DESIGN AND IMPLEMENTATION OF AN OPEN PLATFORM FOR SERVICES INTEGRATION BASED ON MASHUP

Yanlei Liu, Haihong E
PCN & CAD Center, Beijing University of Posts and Telecommunications, Xitucheng Street, Beijing, China

Lin Ma
PCN & CAD Center, Beijing University of Posts and Telecommunications, Beijing, China

Keyword: Mashup, Services integration, REST, JSON, Open platform.

Abstract: In this paper, the integration and fusion technology of the web services(mashup) is firstly introduced, and then we introduce the main technologies used in the building of the platform: RESTful Web Services and JSON which is a text-based lightweight data interchange format. On this basis, we propose an open platform based on mashup for services integration. Based on this open platform the registered users can add external services including the web services and the telecom services, develop and publish mashup processes. Finally, we introduce examples to help understanding the use of the open platform.

1 INTRODUCTION

With the advent of the web 2.0, today's Web is no longer confined to the general sense of browsing and inquiry, but has become a real programming platform. A variety of web 2.0 such as Wiki, Web maps, online shopping, search systems appear in order to meet the different needs of the web users. The attendant problem is that building a web-based application from scratch to meet the various needs is almost impossible. The key to solving these problems is to use the existing Web services and combine them. In the existing web services, the RESTful Web services and their combinations are getting more and more attention (Stefan, 2007).

When the mashup of the internet is in full swing, the personnel in the field of telecommunications have noticed the capacity of mashup which can build and provide business process quickly. Mashup has many advantages
such as business fast provided, high-reuse data sources, low-cost (Duane, 2006). The mobile network with a unique operation and management ability, strong billing support, high viscosity user groups can provide network and services anywhere at any time. We can make the two complement each other after combining them. For telecom operators, in addition to the owner of the communication channels, they still can easily obtain the user's information such as age, income, or even personal preferences (Nilanjan, 2008). All of these information is very valuable to the advertisers. Many operators have done some attempt work in opening their own telecommunications capability.

2 SERVICES AGGREGATION (MASHUP)

2.1 Conception of Mashup

In web development, a mashup is a web page or application that uses and combines data, presentation or functionality from two or more sources to create new services (http://en.wikipedia.org/wiki/Mashup, 2010).

The term implies easy, fast integration, frequently using open APIs (an interface implemented by a software program that enables it to interact with other software) and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data.

DOI: 10.5220/0003587505540559
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (SSE-2011), pages 554-559
ISBN: 978-989-8425-53-9

2.2 Characteristics of Mashup

The main characteristics of the mashup are combination, visualization, and aggregation. Mashup is important to make more useful already existing data, moreover for personal and professional use. To be able to permanently access the data of other services, mashups are generally client applications or hosted online (Martin, 2007).

In the past years, more and more Web applications have published APIs that enable software developers to easily integrate data and functions instead of building them by themselves. Mashups can be considered to have an active role in the evolution of social software and Web 2.0. Mashups composition tools are usually simple enough to be used by end-users. They generally do not require programming skills, they rather support visual wiring of GUI widgets, services and components together (Hazem, 2008). Therefore, these tools contribute to a new vision of the Web, where users are able to contribute. In this paper we propose an open platform for users to build their mashups, besides they can also add their own APIs to the system.

2.3 Architecture of Mashup

The architecture of a mashup is divided into three layers:

- Presentation/ User interaction: this is the user interface of mashups. The technologies used are HTML/XHTML, CSS, JavaScript, Asynchronous JavaScript and Xml (Ajax).
- Web Services: the products functionality can be accessed using the API services. The technologies used are XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.
- Data: Handling the data like sending, storing and receiving. The technologies used are XML, JSON, KML.

Architecturally, there are two styles of mashups: Web-based and server-based. Whereas Web-based mashups typically use the user's Web browser to combine and reformat the data, server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form.

Mashups appear to be a variation of a Facade pattern. That is, it is a software engineering design pattern that provides a simplified interface to a larger body of code (in this case the code to aggregate the different feeds with different APIs).

Mashups can be used with software provided as a service (SaaS).

After several years of standards development, mainstream businesses are starting to adopt Service-oriented Architectures (SOA) to integrate disparate data by making them available as discrete Web services. Web services provide open, standardized protocols to provide a unified means of accessing information from a diverse set of platforms (operating systems, programming languages, applications). These Web services can be reused to provide completely new services and applications within and across organizations, providing business flexibility.

3 KEY TECHNOLOGIES

3.1 RESTful Web Services

Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. The term Representational State Transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation (CHEN, 2008).

A RESTful web service (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of REST. It is a collection of resources, with three defined aspects (http://en.wikipedia.org/wiki/Representational_State_Transfer, 2010):

- The base URI for the web service, such as http://example.com/resources/.
- The MIME type of the data supported by the web service. This is often JSON, XML or YAML but can be any other valid MIME type.
- The set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

The PUT and DELETE methods are idempotent methods. The GET method is a safe method (no side-effect, implies idempotences well).

Unlike SOAP-based web services, there is no "official" standard for RESTful web services. This is because REST is an architecture, unlike SOAP, which is a protocol. Even though REST is not a standard, a RESTful implementation such as the Web can use standards like HTTP, URL, XML, PNG, etc (Indrajit, 2007). Currently, most Web Services providers such as Yahoo, Ebay, Amazon, Google have provided a RESTful Web services APIs, for this reason we provide the API for users to add their RESTful APIs to our open platform.

3.2 JSON

JSON (an acronym for JavaScript Object Notation pronounced) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language -independent, with parsers available for virtually every programming language.

The current network data transmission format are mainly JSON format and the XML format. JSON has its unique advantages to XML (http://en.wikipedia.org/wiki/JSON,2011).

- JSON is much simpler than XML, JSON has a much smaller grammar and maps more directly onto the data structures used in modern programming languages.
- JSON is not extensible because it does not need to be, JSON is not a document markup language, so it is not necessary to define new tags or attributes to represent data in it.
- JSON encoding significantly easier than XML while XML need coding tools.
- JSON's decoding is also easier than XML. XML requires translating the structure of the data into a document structure. This mapping can be complicated. JSON structures are based on arrays and records. That is what data is made of.

JSON is just beginning to become known. Its simplicity and the ease of converting XML to JSON makes JSON ultimately more adoptable. And in the open platform, we use the JSON to format the data between the internal data exchange. (CHEN, 2009)

4 DESIGN OF THE PLATFORM

4.1 Architecture of the Platform

Figure 1 illustrates the architecture of the open platform.

The open platform can be divided into three parts: the service access platform, the process development platform, the process publishing platform. The service access platform is responsible for users to add their own service APIs which consists mainly of telecommunications service APIs and the Internet service APIs. The process development platform based on the service APIs added through the service access platform by the registered users. Registered users can build their mashup processes on this platform. The process publishing platform is responsible for generating JavaScript or HTML code which can be embed in web pages for one mashup process. The data center is to save the data information generated by the open platform. Users login in and out through the upper unified portal.

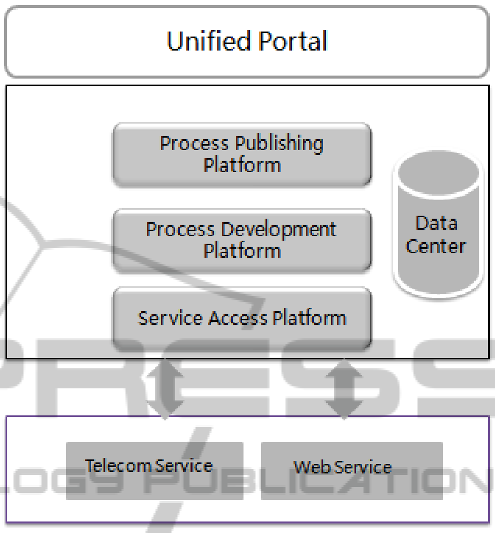


Figure 1: The Architecture of the Platform.

4.2 Interface Design

4.2.1 Service Access Interface

The service access interface is responsible for the access of external service APIs which mainly include the Web service APIs and the telecommunication service APIs. In this platform we only accept the RESTful APIs. Users who want to add external APIs login in the service
access platform fill in the form with corresponding parameters. After submitting the form you will see the service appears at the process development platform.

4.2.2 Data Access Interface

This interface give the implementation of the communication between our server and the Web services. The users send their request to our platform, and then the platform send the request to the corresponding Web services. We design a class to visit APIs throw the POST method and the GET method.

4.2.3 Data Analysis Interface

This interface helps to analyze the response text from the Web service APIs. We convert the response data into the format of JSON and then send the result to the user who send the request.

4.2.4 Process Publishing Interface

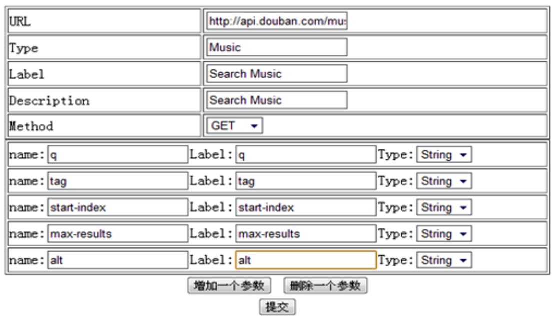
The mashup process generated from the process development platform could not be used directly, the process must be convert into the JavaScript or HTML code which can be embedded into web pages. Therefore we design a special module to help the users convert their process into JavaScript or HTML code.

5 INSTANCE DEVELOPMENT

5.1 Service Access

The system is an open platform means that the user can add external service APIs to the platform effectively and easily. Now we take the DouBan's open API for music search for example to show how to add external APIs to the open platform. The one who want to use this platform must have registered at the system. In order to distinguish the different APIs added by different users, we build a second directory for each user to show their own APIs, the current user can view and use the service APIs other users add.

First, users login in the system and open the page for API access, figure2 shows the page. After submitting the form, figure3 shows the result, you will say the API in the process development platform and you can add this service to your mushup process.



URL	http://api.douban.com/mu:
Type	Music
Label	Search Music
Description	Search Music
Method	GET

name: q	Label: q	Type: String
name: tag	Label: tag	Type: String
name: start-index	Label: start-index	Type: String
name: max-results	Label: max-results	Type: String
name: alt	Label: alt	Type: String

Figure 2: The Page for Service.

Access

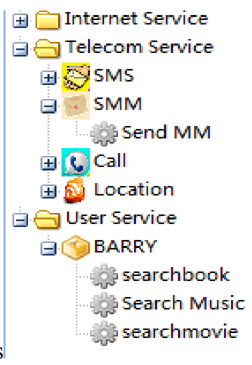


Figure 3: The Services Package.

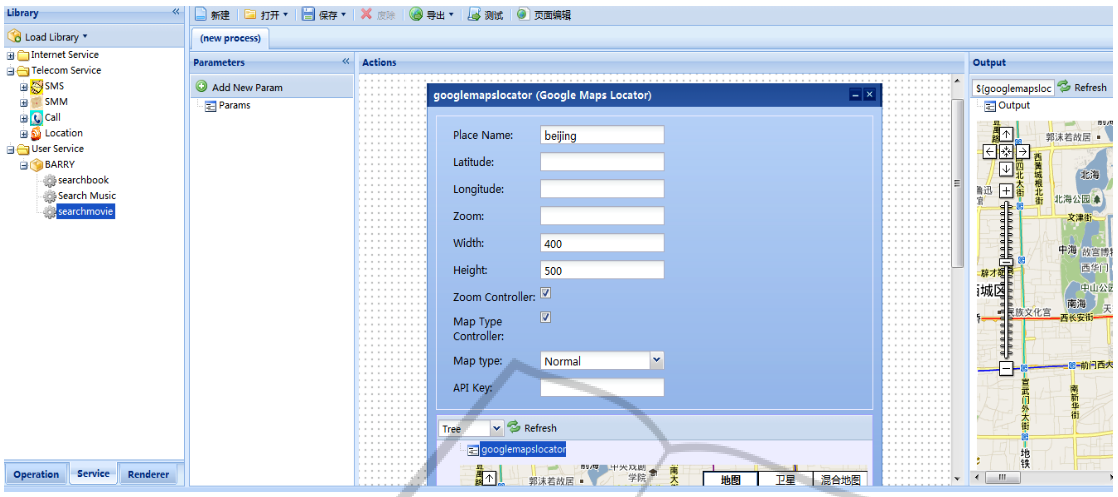
5.2 Build One Mashup Process

Users should also login in the process development platform to build their mashups. The interface is divided into three parts as shown in figure4:

- Library panel: in this panel appears the operators, services and renderers that can be used to build the mashup:
  - ➢ Operators:The information obtained can be manipulated with the operators . For example, it's possible to sort, filter or group the information by the parameter chosen.
  - ➢ Service: Several RESTful services that can be invoked.Which contains three kinds of APIs:The Web Services APIs,the APIs based on the services of the telecommunication class,the APIs that the users add.
  - ➢ Renderers: the information can be represented in different renders.
- Actions panel: the operators can be drag and drop to it and is where they are combined in order to collaborate to form a mashup. All the operators, services and renderers are represented in the panel as a form with some fields, this is the input of the operator .Submitting the form, the result of the operator is shown in the below part of the form.
- Output panel: the mashup should be dragged and dropped to this panel to be exported.

5.3 Publish Mashup Process

After building the mashup process, the user can export the process into JavaScript or HTML code so that the developer can embed these code in their applications. Figure4 is the example to use the Google Maps API, if you click the "Publish as Javascript", you will get the Javascript code as follows:



Place Name: beijing
Latitude:
Longitude:
Zoom:
Width: 400
Height: 500
Zoom Controller: ✓
Map Type Controller: ✓
Map type: Normal
API Key:

Figure 4: The Page for Process Development.

```
<script type="text/javascript"
src="http://59.64.154.86:8080/MyCocktai
l/js/afrous/afrous-config.js"></script>

<script type="text/javascript"
src="http://
59.64.154.86:8080/MyCocktail/js/afrous/
afrous-core.js"></script>

<script type="text/javascript"
src="http://
59.64.154.86:8080/MyCocktail/js/afrous/
afrous-package.js"></script>

<script type="text/javascript"
src="http://
59.64.154.86:8080/MyCocktail/js/afrous/
renderers/google-maps.js"></script>

<script type="text/javascript">

  var procdef = new
afrous.ProcessDef({"name":"","descripti
on":"","params":[],"output":"${googlema
pslocator}","actions":[{"type":"Rendere
r.GoogleMaps.GoogleMapsLocator","name":
"googlemapslocator","inputs":{"placeNam
e":"beijing","width":"400","height":"50
0","zoomController":"true","mapTypeCont
roller":"true","mapType":"Normal"}}]});

  var proc = new
afrous.ProcessInstance(procdef);

  proc.start(function(result)
{   result.render(document.getElementBy
Id("mydiv"));

  })

</script>
```

6 CONCLUSIONS

As a new member of the Web 2.0, the open platform for building mashups is the exploration stage of development. In this paper, we attempt to put forward a fusion of an open platform for services integration, introduce the key interfaces designed in the system and teach the readers how to use the platform through a complete example.

Practice shows that the platform can accept the external services effectively and easily and suits mashup process development perfectly, the developing time is also shorted.

ACKNOWLEDGEMENTS

This work is supported by the National Key project of Scientific and Technical Supporting Programs of China (Grant Nos.2008BAH24B04, 2008BAH21B03, 2009BAH39B03); the National Natural Science Foundation of China(Grant No.61072060); the Program for New Century Excellent Talents in University (No.NECET-08-0738); and the Innovation Technology Star Program of Beijing (Grant No.2007A045).

REFERENCES

Stefan Tilkov, A Brief Introduction to REST,2007.12.25. http://blog.csdn.net/niusi123/archive/2007/12/11/1929744.aspx.

Duane Merrill. Mashups: *The new breed of Web app. IBM Developerworks,* 2006, http://www.ibm.com/developerworks/xml/ library/x-mashups.html?S_TACT=105AGX52&S_CMP=cn-a-x.

WiKipedia Mashup Concept [EB/OL]. [2010-11 -20]. http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid).

Martin Brown. Plans for the rich Web application backplane.IBM Developworks,2007,http://www.ibm.com/developerworks/xml/library/x-backplane/index.html?S_TACT=105AGX52&S_CMP=cn-a-x.

Nilanjan Banerjee, Koustuv Dasgupta.Telecom Mashups: enabling Web2.0 for telecom services. In: *Proceedings of the 2nd International Conference,* Suwon,Korea,2008.

Hazem Elmeleegy, Anca Ivan,Rama Akkiraju, et al. Mashup advisor: a recommendation tool for Mashup development. In: *ICWS'08, IEEE International Conference*, USA, 2008.

Chen, Liang, TAO, Hong-cai, Research and application of Mashup based on REST-style, *Journal of Chengdu University of Information Technology*, Vol.23 No.5 Dct. 2008.

WiKipedia REST Concept [EB/OL]. [2010-11-20]. http://en.wikipedia.org/wiki/Representational_State_Transfer

Alex Rodriguez, Software Engineer, IBM. *RESTful Web services: The basics,* 2008.12.22. http://www.ibm.com/developerworks/webservices/library/ws-restful/?S_TACT=105AGX52&S_CMP=content.

Indrajit Poddar, Zhi Gan, Yue Lin Liu, Software Engineer, IBM, Building SOA composite business services, Part 3: Build consumable *Web Services using the RESTarchitectural style in WebSphere,* 2007.5.12.

WiKipedia JSON Concept [EB/OL]. [2010-11-20]. http://en.wikipedia.org/wiki/JSON.

Chen, Zhu, Dai, Adie, Wang Yue-fen. Application of JSON to Mashup Web Service. *Land and Resources Informatization.*Vol.24 No.5 2009.