Doing Webservices Composition by Content-based Mashup

Example of a Web-based Simulator for Itinerary Planning

Mohamadou Nassourou
Department of Computer Philology & Modern German Literature
University of Würzburg
Am Hubland
D - 97074 Würzburg
mohamadou.nassourou@uni-wuerzburg.de

Abstract: *Webservices composition is traditionally carried out using composition technologies such* as *Business Process Execution Language (BPEL) [1] and Web Service Choreography Interface (WSCI) [2]. The composition technology involves the process of web service discovery, invocation, and composition. However these technologies are not easy and flexible enough because they are mainly developer-centric. Moreover majority of websites have not yet embarked into the world of web service, although they have very important and useful information to offer. Is it because they have not understood the usefulness of web services or is it because of the costs? Whatever might be the answers to these questions, time and money are definitely required in order to create and offer web services. To avoid these expenditures, wrappers [7] to automatically generate webservices from websites would be a cheaper and easier solution. Mashups offer a different way of doing webservices composition. In web environment a Mashup is a web application that brings together data from several sources using webservices, APIs, wrappers and so on, in order to create entirely a new application that was not provided before.*
This paper presents first an overview of Mashups and the process of web service invocation and composition based on Mashup, then describes an example of a web-based simulator for navigation system in Germany.

Keywords: Mashup, Webservice Composition, Wrappers

Introduction

A Mashup is a website or an application which integrates data from several sources in order to provide a more useful and concise service. They create composite web applications in an ad hoc manner.
Mashups retrieve data from content providers through APIs/Webservices, Web feeds (RSS,Atom) and Screen Scraping. Originally they were used in music for combining tracks from different artists. A Mashup could be viewed as a software engineering design pattern that provides a simplified interface to a larger body of code.

Majority of websites have not embarked into the world of web service although they have very important and useful information to offer. Is it because they have not understood the usefulness of web services or is it because of the costs?

Whatever the answers to these questions might be, time and money are definitely required in order to create and offer web services. To avoid these expenditures, wrappers [3] to automatically generate webservices from websites would be a cheaper and easier solution.

In fact offering a webservice implies simply providing data that could be automatically processed by a machine. Every XML document is automatically processable by any machine. So the information that is dedicated to human consumption could be processed by machines as well, if it is formatted as XML. Wrappers to extract relevant information are a good solution for clients.

Background

A Mashup is a web application that integrates data from several sources into a single completely new service. A Mashup site is supposed to access data from other sites and process that data in order to add its value. They retrieve data from content providers through APIs/Webservices, Web feeds, and screen scraping technique.

Content aggregation has been around decades ago. Traditional dynamic web applications are usually made of aggregated contents. Portals are an example of older content integration technique. They use portlets to generate information that are combined into a single web page. The portlets could be locally hosted or remotely accessed from another server. However portals combine information basically on the server side. The aggregated contents are presented separately without overlapping and remodeling.

Mashups to the contrary offer a more robust and sophisticated content aggregation both on the client side as well as on the server side. In other words they present completely new structured hybrid contents.

Architecture of a Mashup

In web environment a Mashup application comprises three parts: Content Providers, Mashup site, and Client's Web Browser.

Figure.1 shows a concise architecture of a Mashup

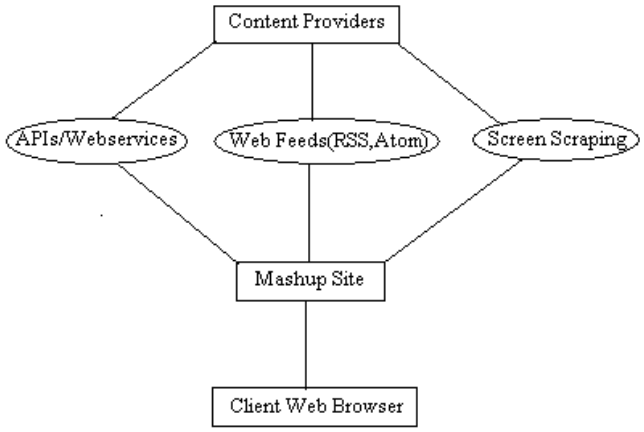

Figure.1 Architecture of a Mashup

Content Providers expose their services through APIs/Webservices, Web feeds, or HTML pages. Then a Mashup Site calls the relevant API or develops Screen Scraping method such as wrappers to retrieve needed data, which are either integrated on the server or sent to Client's Web Browser for mashing and presenting.

a) Content Providers

They offer data that client browsers need to retrieve and combine. Data are obtained using API/webservice or provider's web pages contents (HTML documents).

b) Mashup Site.

This is the site which hosts the Mashups. Often Mashups use a combination of both server and client-side programming languages (e.g PHP, Javascript) to perform their data aggregation.

c) Client's Web Browser

This is the user interface to the Mashup. Several Mashups use client-side programming languages (e.g Javascript) to combine and compose their contents.

Types of Mashup

Basically there are three types of Mashup.

a) Presentation-level Mashups
These are based on web clipping technique. They make use of existing web application interfaces.

b) Logic-based Mashups
These are Mashups which borrow functionality of any web-enabled application.

c) Content-based Mashups
Content-based Mashups rely mostly on screen scraping technique to assemble data from any web-enabled data sources.

Mashup Operators

In a RESTful [10] style webservice, the process of creating a Mashup involves the following operations:

a) **Retrieve** representation of some resources.
b) **Filter** those representations.
c) **Pipe** the filtered representations to next steps.
d) **Compose** / Combine the filtered representations in different models.
e) **Select** one model of the composition results.
f) **View** the selected model.

In general a Mashup connects to several websites, retrieves representations of some resources, filters each one of them. Then it combines them in different ways. In fact web service composition theory states that the output of one service is used as the input to another one. After that it selects the appropriate composition according to its defined objectives. Finally it presents the result to the client's web browser or save it in any relevant format for further use.

A Practical Web-based Simulator for Itinerary Planning in Germany

This section explains how to create a novel and useful web service out of the following services: Google Map API [4], YellowMap web service [5], and RVM (Rhein-Main-Verkehrsverbund) web site [6]. The created web service is invoked by the simulator which displays the result as shown in Fig. 2.

Fig. 2 shows the graphical user interface of the simulator with a sample result.

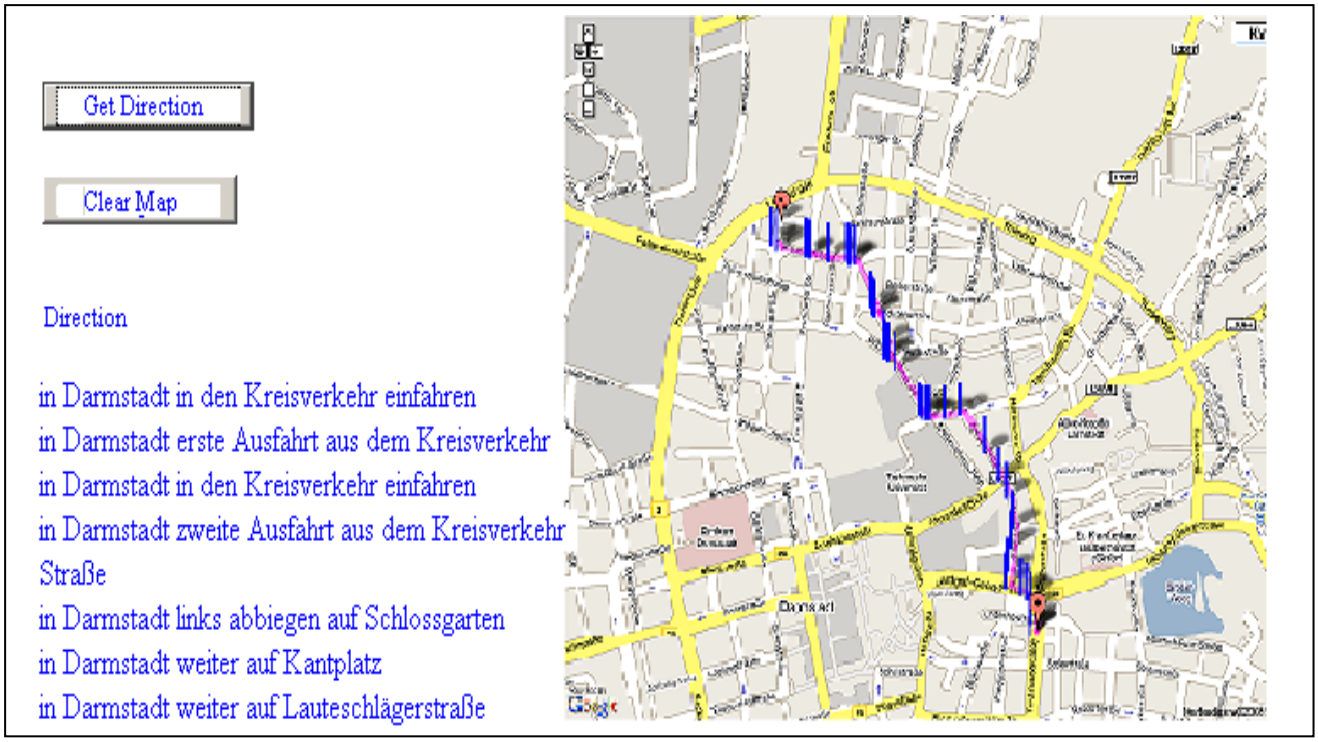

Fig.2 Graphical User Interface of the Simulator

Google Map APIs offer many operations. I was mainly interested in displaying the transportation connections from departure to destination localities on the map. GoogleMap is displayed within an HTML document. To use the map a key which is obtained through online registration at [7] is required. Once the key is received the following sample Javascript code shows how to call the map and customize it:

```
<html>
<head>
<script
src='http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAPOtupwlC6PfGjGrTwq9ykxTpRofnNLHdS2OYZupVNjxiHHlQMhQzubJ_5N1qnGKX05tjUKJbLHAv4w'
type='text/javascript'>
</script>
<script type='text/javascript'>
var map = new
GMap(document.getElementById('map_canvas'));
map.centerAndZoom(new GPoint(8.660002, 49.878916), 2);
map.addControl(new GSmallMapControl());
map.addControl(new GMapTypeControl());
</script>
</head>
<body>
<div id='map_canvas' style='width: 1000px; height:
700px'></div>
</body>
</html>
```

The retrieval of the coordinates of a given location from Google Map API was done with the following sample PHP program:

```
$station = "Darmstadt Hauptbahnhof";
$url =
"http://maps.google.com/maps/geo?q=$station&output=csv&key=ABQIAAAAPOtupwlC6PfGjGrTwq9ykxTpRofnNLHdS2OYZupVNjxiHHlQMhQzubJ_5N1qnGKX05tjUKJbLHAv4w
";
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_VERBOSE, 1);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$response = curl_exec($ch);
curl_close($ch);
$data = explode(",", $response);
$latitude = $data[3];
$longitude = $data[2];
```

The connection and query are made using CURL library [11].

From YellowMap's web service, the address of a location given its coordinates, and pedestrian route are obtained.
The RMV web site provides full transportation connections from departure to destination localities.
However it does not offer pedestrian route direction.
To retrieve the needed information from RMV web site I wrote a wrapper which has been explained in [8].

Briefly the wrapper operates as follows:
A query with departure and destination stations as parameters was sent to RMV web site. Then the web site returned a result in HTML form containing the route directions. After parsing the page, relevant information such as departure station, arrival station, departure time, arrival time, locomotives (bus, tram, train, pedestrian) were obtained.
The input to GoogleMap web service is the stations names that were returned by the RMV web site.
So invoking GoogleMap webservice returned the coordinates of each station.

YellowMap web service was used to obtain the pedestrian route direction. By supplying the coordinates of the start and end of pedestrian stations that were returned by GoogleMap, YellowMap offered street names and junctions that constitute the required pedestrian route.
Finally the itinerary was generated by substituting the pedestrian phase of RMV's response with the YellowMap's pedestrian route direction.
After that the result was formatted as XML document which was used as a completely new web service in turn.

Discussion

The invocation of YellowMap web service was not as simple as expected because of lack of sufficient documentation. Sample codes to illustrate the usage of each operation are missing.

According to web service composition theory, the output of one service is used as the input to another one. However this is not a straight forward process. In fact several problems were encountered such as schema mapping between data types, large data, as well as data quality problems.

For instance some addresses returned by the YellowMap service were unrecognized by the RMV service and vice versa. YellowMap and RMV services were responding with different addresses for some given coordinates.

Quite often the stations names retrieved from RMV website had to be processed before passing them to Google Map web service in order to retrieve their coordinates.

Conclusion

This research shows that Mashups are viable alternatives to traditional webservices creation and composition. An overview of Mashup technology has been presented. How to use existing web technologies such as PHP, Javascript, Google Map API, YellowMap webservice, and RMV website for developing a practical web-based simulator for itinerary planning in Germany has also been described.

References

[1] http://en.wikipedia.org/wiki/Business_Process_Execution_Language
[2] http://www.w3.org/TR/wsci/
[3] http://code.google.com/apis/maps/index.
[4] http://www.yellowmap.de/
[5] http://www.rmv.de/
[6] http://code.google.com/apis/maps/signup.html
[7] Mohamadou Nassourou (2010), "Empirical Study on Screen Scraping Web Service Creation Case of Rhein-Main-Verkehrsverbund (RMV)".
[8] www.php.net/
[9] http://en.wikipedia.org/wiki/JavaScript
[10] http://en.wikipedia.org/wiki/Representational_State_Transfer
[11] http://php.net/manual/en/book.curl.php