Lash-Ups: A Toolkit for Location-Aware Mash-Ups

Joel Brandt, Scott R. Klemmer
Stanford University HCI Group
Computer Science Department
Stanford, CA 94305-9035, USA
{jbrandt, srk}@cs.stanford.edu

ABSTRACT

Mash-ups of web content are a quickly growing in popularity due to an increasing amount of public APIs for web services. Similarly, web-enabled cellular phones are becoming increasingly pervasive. We present Lash-Ups, a toolkit that enables programmers to create mash-up-like location-aware web services for phones. This toolkit solves two fundamental problems. First, it provides a simple, standard API for programmers to use locations of users as an input to their Lash-Ups. Second, it provides a way for programmers to distribute their Lash-Ups to users based on location. We support both push- and pull-based applications, and our platform runs on commodity GSM cellular phones.

ACM Classification: H5.2 [Information Interfaces and Presentation]: User Interfaces – *Input devices and strategies, interaction styles*. D2.2 [Software Engineering]: Design Tools and Techniques – *Software libraries*.

General terms: Design, Experimentation, Standardization

Keywords: Location-aware, toolkit, mobile, end-user programming, mash-up, example modification

INTRODUCTION

Web-service mash-ups are enjoying significant popularity on the Internet. In surveying current mash-ups, it is easily seen that the majority of mash-ups accomplish a straightforward task: information that is available from a variety of sources is combined in a form that is more convenient to the end user.

We suggest that a large class of location-aware applications fit this model as well: the information a user wants is already available on the web, but it is not easily accessed in a mobile setting. Currently, writing location-aware mash-ups is difficult for two reasons: first, there is no straightforward method of enabling a user's location to be part of the mash-up's inputs, and second, there is no common way to distribute mash-ups to users in locations where the mash-up is most applicable. We introduce Lash-Ups, a toolkit for location-aware mash-ups that addresses both of these issues.

THE LASH-UP TOOLKIT

The Lash-Up toolkit consists of two components. The *Lash-Up Server* provides a common place for Lash-Ups to be distributed to users, and provides a common API for Lash-Ups to access location information of their users. The *Lash-Up Client* is a light-weight application that runs on users' phones, gathering location information for communication to the Lash-Up Server.

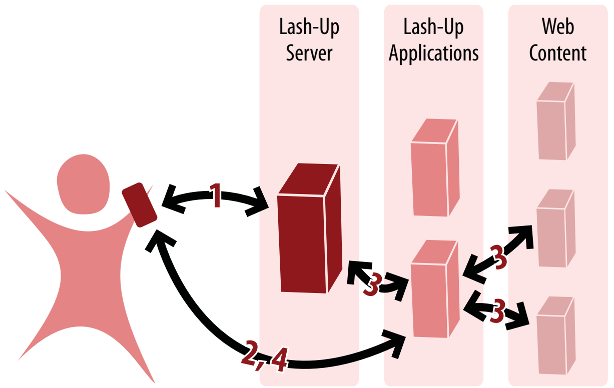

Figure 1. The Lash-Up system interaction for pull-based content. (1) The user accesses the Lash-Up Server to retrieve a list of relevant Lash-Ups. (2) The user chooses an application to access. (3) The application communicates with the Lash-Up Server as well as other web content. (4) The Lash-Up content is delivered to the user.

The Lash-Up Client

The Lash-Up Client is designed with two ideas in mind: the client must require minimal interaction for the user to get the information she desires and the client must run on commodity cell phones. To meet the second requirement, we chose to use GSM cell tower information as the location data [1], and to use a web browser as the content delivery and UI mechanism. To meet the first requirement, we designed a system where location is the principle means of information access.

Our client is a light-weight application that gathers location information and places it in an HTTP request to the Lash-Up Server. The Lash-Up Server builds an HTML page that lists relevant Lash-Ups. The links to these Lash-Ups are constructed as GET requests with the user's location information embedded in the URL. In this way, when the user selects a Lash-Up to access, the location information is forwarded to the Lash-Up. Thus, accessing information relevant to the user's location becomes a simple three-step process: *click* the Lash-Up Client, *read* the list of relevant Lash-Ups, and *choose* the item of interest.

The Lash-Up Server

The Lash-Up Server accomplishes two related tasks: providing information about relevant Lash-Ups to the end user, and providing information about the location of the user to the Lash-Up itself. In order to accomplish these goals, the server must maintain a repository of available Lash-Ups, and information about where they are relevant. Developers of Lash-Ups register their Lash-Up with the server. At this time, the developer provides information about locations where the application is relevant.

Li *et al*. present a taxonomy of location-aware applications [3]. Within this taxonomy, we see three different types of location information being used: information about an *exact* position such as a user's latitude and longitude coordinates; information about the *semantics* of the location, such as "supermarket or "train station"; and information expressing the *relationship* between entities, such as the distance between two people.

Our server supports all three types of location information, and translates the user's exact coordinates to the appropriate type for the application. *Exact* location information is acquired through a mapping of GSM towers to latitude and longitude, as done in the Place Lab project [2]. *Semantic* information is acquired through a tagging approach. Users or programmers may label locations (GSM towers) with tags, such as "supermarket". Then, if an application is registered as being relevant to a particular tag, it will show up in a user's list whenever she is at a location (GSM tower) tagged as such. *Relationship* information is accessed via the framework supporting push-based applications described below. A Lash-Up may query the server for the most recently-recorded location of any user.

We note that we have not addressed how we insure that quality content is delivered in such a system. We propose a technique similar to social bookmarking in our Future Work section.

While space prohibits a detailed description here, the full API documentation can be viewed at http://hci.stanford.edu/lash-ups.

Supporting Push-Based Applications

In addition to the pull-based Lash-Up access described above, the client can be set to periodically update the Lash-Up Server with its current location, so that push-based applications can be provided, such as a location-aware reminder service [4]. Lash-Up applications can register a *callback URL* with the Lash-Up Server, tied to a particular user and location. When the user enters this location, the Lash-Up Server accesses this callback URL. The Lash-Up then performs the pertinent action, such as sending the user an SMS message reminding her to buy milk. While privacy is not the main focus of our work, we note that if privacy is a concern for users, the automatic updating may be disabled. Thus, location information would be disclosed only at the user's discretion. Additionally, techniques presented by Tang *et al*. could easily be adapted to our system [5].

LASH-UP EXAMPLES

In the accompanying poster and video, we show the development of two simple Lash-Ups, and the use of a third. The first application leverages the location-based content distribution of Lash-Ups to deliver a static webpage with visitor information. Almost no programming is necessary to develop this application.

The second application combines location information gathered from the Lash-Up API with an existing web search application. With only two button presses, the user is able to retrieve a list of coffee shops near her current location. Again, the development of this application requires only a few lines of code.

The third application demonstrates the use of the Lash-Up API's push-based features. It allows the user to submit a reminder that will be sent to him via SMS whenever he is at a particular location.

WHY LASH-UPS MAKE SENSE

Lash-Ups are an attractive option for many location-aware applications for four primary reasons. First, we suggest location-aware computing is more likely to succeed through a "killer ecology" of many small applications rather than a few large applications. Second, these applications need to be kept up to date, which is more manageable in a user-supported development community. Third, many applications simply consist of providing pre-existing information in a useful location, which matches the mash-up paradigm perfectly. Finally, while writing handset applications is comparatively difficult, web delivery of applications is straightforward and ubiquitous, and the programming model is familiar to a far greater number of programmers.

FUTURE WORK

We are currently exploring tools to help programmers more quickly and easily prototype and develop Lash-Ups. Additionally, we are investigating mechanisms for insuring that quality content is delivered to users, and that relevant semantic information is entered into the system. We are planning to leverage techniques similar to social bookmarking, which uses group behavior to identify quality content. Finally, we are exploring the use of additional attributes of the user's context—such as time of day and personal preferences—in Lash-Up distribution.

REFERENCES

1 Borriello, G., M. Chalmers, A. LaMarca, and P. Nixon. Delivering real-world ubiquitous location systems. *Communications of the ACM* 48(3). pp. 36-41, 2005.

2 LaMarca, A., Y. Chawathe, S. Consolvo, *et al.*, Place Lab: Device Positioning Using Radio Beacons in the Wild, in *Pervasive*. 2005.

3 Li, Y., J. I. Hong, and J. A. Landay, Topiary: a tool for prototyping location-enhanced applications, in *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 2004.

4 Sohn, T., K. A. Li, G. Lee, *et al.*, Place-Its: A Study of Location-Based Reminders on Mobile Phones, in *Ubiquitous Computing*. 2005.

5 Tang, K. P., P. Keyani, J. Fogarty, and J. I. Hong, Putting People in their Place: An Anonymous and Privacy-Sensitive Approach to Collecting Sensed Data in Location-Based Applications, in *ACM Conference on Human Factors in Computing Systems (CHI)*. 2006.