Developing Client-Side Mashups: Experiences, Guidelines and Reference Architecture

Arto Salminen, Department of Software Systems, Tampere University of Technology, Tampere, Finland
Tommi Mikkonen, Department of Software Systems, Tampere University of Technology, Tampere, Finland
Feetu Nyrhinen, Institut für Angewandte Systemtechnik Bremen GmbH, Bremen, Germany
Antero Taivalsaari, Nokia Research Center, Helsinki, Finland

ABSTRACT
Software mashups that combine content from multiple web sites to an integrated experience are a popular trend. However, methods, tools and architectures for creating mashups are still rather undeveloped, and there is little engineering support behind them. In this paper the authors present guidelines that can serve as a helpful starting point for the design of new mashups. Guidelines focus mainly on mashup creation methods. Furthermore, they describe a reference architecture for client-side mashup development. In addition, the authors provide insight into mashup development based on their practical experiences in implementing various sample client-side mashup applications and tools for creating them. The long term goal of the authors’ work is to facilitate the development of compelling, robust and maintainable mashup applications, and more generally ease the transition towards web-based software development.

Keywords: Mashup Development, Mashups, Web Applications, Web-Based Software Development, Web Engineering

INTRODUCTION
Software mashups that combine content from multiple web sites are a hot trend. It is becoming increasingly common to find compelling web applications that aggregate and deliver images, text, and other data from numerous web sites in an innovative and often entirely unforeseen fashion. The ability to combine code and content from multiple sources from anywhere in the world has opened up entirely new possibilities for software development. The trend towards software mashups has given rise to a number of environments and tools that have the specific objective to make mashup development easier. However, due to the relatively ad hoc nature of mashups, it has been – and still is – difficult to provide developers with general purpose tools for mashup development. Furthermore, there are various restrictions and technical limitations that arise from the design of the web browser, such as the same-origin principle (Rudeman,
that prevents a web client from easily downloading data from multiple web sites. Common file formats such as XML and JSON (JavaScript Object Notation), techniques such as RESTful web services (Fielding & Taylor, 2002), and JavaScript libraries such as Dojo (http://www.dojotoolkit.org/), jQuery (http://jquery.com/) and Scriptaculous (http://script.aculo.us/) have turned out to be invaluable in mashup development, though.

Both server-side and client-side mashups can be implemented. Today, most mashup development tools (see a summary of the tools in Nyrhinen, Salminen, Mikkonen, & Taivalsaari, 2009; Taivalsaari, 2009) are intended for server-side use, that is, the downloading, processing and generation of web content is performed on the server. In client-side mashups, in contrast, the downloading and combination of web content is performed on the client (e.g., in a web browser running on a desktop computer), typically utilizing the JavaScript language (Flanagan, 2006; Crockford, 2008) and additional libraries to implement the application.

In this paper we examine technical development of client-side mashups and the characteristics of client-side mashups in general. Based on our hands-on experiences in developing various client-side mashups and tools, we provide a novel set of guidelines that can help a developer choose the right methods when building new mashups. Even though mashup patterns for enterprise server-side mashups have been described (Ogrinz, 2009), this kind of a set of practical methods for creating client-side mashups has not been available earlier. Furthermore, in order to enable different vendors to provide service interfaces and to compose mashup clients, we describe a reference architecture for client-side mashups. In summary, the goal of the paper is to give an extended overview of our experiences on client-side mashup development. This paper is an extended version of our earlier papers (Salminen, Nyrhinen, Mikkonen, & Taivalsaari, 2010; Mikkonen & Salminen, 2011).

The paper is structured as follows. First, we discuss mashups and mashup development in general. Then, we group our findings into practical guidelines that can be applied to the development of new mashups. Afterwards, we describe our mashup reference architecture. Following that, we provide hands-on mashup development examples based on real-life applications that we have created. We then go on to discuss the experiences and lessons learned during the implementation of these applications. Finally, we then draw some conclusion and discuss the directions for future work, including the steps that pave the way towards mashware (Taivalsaari, 2009) – full-fledged mashup applications that consist of software components that have been downloaded from multiple sites and then dynamically combined into new applications.

BACKGROUND AND RELATED WORK

A typical software mashup combines data, images, code, and other content from multiple sources into a new user experience. In today’s mashups, data and content are downloaded most commonly from different web sites, but in principle data from any source (such as the user’s local computer or an intranet database) can be used. In addition to aggregating data from multiple sources, mashup applications typically provide an alternative user interface or add advanced filtering or visualization capabilities to a web service.

An important distinction exists between portals and mashups. Portals are web pages that contain information that is retrieved from different sources. Usually the user interface of a portal consists of numerous “portlets”, separate pieces of content that are presented in unified way. However, in contrast to mashups these portlets are isolated from each other, and cannot communicate with each other. Mashups are more integrated applications that are constructed in such a fashion that the user typically cannot distinguish data origins. Furthermore, mashups usually create entirely new visualizations from the information they are based on instead of just aggregating the information into a single view.
Related Content

Fuzzy Dynamic Load Balancing in Virtualized Data Centers of SaaS Cloud Provider

Agile Workflow Technology and Case-Based Change Reuse for Long-Term Processes
[www.igi-global.com/article/agile-workflow-technology-case-based/2431?camid=4v1a](www.igi-global.com/article/agile-workflow-technology-case-based/2431?camid=4v1a)
Memes and Mutation: Societal Implications of Evolutionary Agents in Push Technologies
www.igi-global.com/article/memes-mutation-societal-implications-evolutionary/2377?camid=4v1a

www.igi-global.com/chapter/cross-layer-cooperative-protocol-for-industrial-wireless-sensor-network/173245?camid=4v1a