Mashups： Web应用新趋势

Mashups简介

Mashups are an exciting genre of interactive Web applications that draw upon（利用） content（内容，目录） retrieved（检索） from external data sources to create entirely new and innovative services. They are a hallmark（检验印记，标志） of the second generation of Web applications informally（通俗的，非正式的） known as Web 2.0. This introductory（引导的，介绍的） article explores what it means to be a mashup, the different classes of popular mashups constructed today, and the enabling technologies that mashup developers leverage to create their applications. Additionally, you'll see many of the emerging technical and social challenges that mashup developers face.

Mashups是一个令人振奋的交互Web应用类型，其利用外部数据源目录检索从而创建崭新的创新性的服务。他们是第二代Web应用（通俗的说就是Web 2.0）的标志。这篇引导性的文章向我们挖掘了什么是mashup，当前流行的mushup构造由哪些不同的类组成，以及mashup开发者通常创建他们的应用可能用到的技术。另外，你也将发现很多mashup开发人员面对的新兴技术和社会挑战。

## Introduction

A new breed of Web-based data integration applications is sprouting up all across the Internet. Colloquially termed mashups, their popularity stems from the emphasis on interactive user participation and the monster-of-Frankenstein-like manner in which they aggregate and stitch together third-party data. The sprouting metaphor is a reasonable one; a mashup Web site is characterized by the way in which it spreads roots across the Web, drawing upon content and functionality retrieved from data sources that lay outside of its organizational boundaries.

This vague data-integration definition of a mashup certainly isn't a rigorous one. A good insight as to what makes a mashup is to look at the etymology of the term: it was borrowed from the pop music scene, where a mashup is a new song that is mixed from the vocal and instrumental tracks from two different source songs (usually belonging to different genres). Like these "bastard pop" songs, a mashup is an unusual or innovative composition of content (often from unrelated data sources), made for human (rather than computerized) consumption.

So, what might a mashup look like? The ChicagoCrime.org Web site is a great intuitive example of what's called a mapping mashup. One of the first mashups to gain widespread popularity in the press, the Web site mashes crime data from the Chicago Police Department's online database with cartography from Google Maps. Users can interact with the mashup site, such as instructing it to graphically display a map containing pushpins that reveal the details of all recent burglary crimes in South Chicago. The concept and the presentation are simple, and the composition of crime and map data is visually powerful.

In Mashup genres, you'll survey the popular genres of mashups, including mapping mashups. Related technologies overviews the technology landscape that relates to the construction and operation of mashups. Technical challenges and Social challenges present the eminent technical and social challenges, respectively, affecting mashups.

## Mashup genres

In this section, I give a brief survey of the prominent mashup genres.

### Mapping mashups

In this age of information technology, humans are collecting a prodigious amount of data about things and activities, both of which are wont to be annotated with locations. All of these diverse data sets that contain location data are just screaming to be presented graphically using maps. One of the big catalysts for the advent of mashups was Google's introduction of its Google Maps API. This opened the floodgates, allowing Web developers (plus hobbyists, tinkerers, and others) to mash all sorts of data (everything from nuclear disasters to Boston's CowParade cows) onto maps. Not to be left out, APIs from Microsoft (Virtual Earth), Yahoo (Yahoo Maps), and AOL (MapQuest) shortly followed.

Video and photo mashups

The emergence of photo hosting and social networking sites like Flickr with APIs that expose photo sharing has led to a variety of interesting mashups. Because these content providers have metadata associated with the images they host (such as who took the picture, what it is a picture of, where and when it was taken, and more), mashup designers can mash photos with other information that can be associated with the metadata. For example, a mashup might analyze song or poetry lyrics and create a mosaic or collage of relevant photos, or display social networking graphs based upon common photo metadata (subject, timestamp, and other metadata.). Yet another example might take as input a Web site (such as a news site like CNN) and render the text in photos by matching tagged photos to words from the news.

Search and Shopping mashups

Search and shopping mashups have existed long before the term mashup was coined. Before the days of Web APIs, comparative shopping tools such as BizRate, PriceGrabber, MySimon, and Google's Froogle used combinations of business-to-business (b2b) technologies or screen-scraping to aggregate comparative price data. To facilitate mashups and other interesting Web applications, consumer marketplaces such as eBay and Amazon have released APIs for programmatically accessing their content.

News mashups

News sources (such as the New York Times, the BBC, or Reuters) have used syndication technologies like RSS and Atom (described in the next section) since 2002 to disseminate news feeds related to various topics. Syndication feed mashups can aggregate a user's feeds and present them over the Web, creating a personalized newspaper that caters to the reader's particular interests. An example is Diggdot.us, which combines feeds from the techie-oriented news sources Digg.com, Slashdot.org, and Del.icio.us.

Related technologies

This section gives an overview of the technologies that are facilitating the development of mashups. For further information about any of these technologies, consult Resources at the end of this article. The architecture A mashup application is architecturally comprised of three different participants that are logically and physically disjoint (they are likely separated by both network and organizational boundaries): API/content providers, the mashup site, and the client's Web browser.

• The API/content providers. These are the (sometimes unwitting) providers of the content being mashed. In the ChicagoCrime.org mashup example, the providers are Google and the Chicago Police Department. To facilitate data retrieval, providers often expose their content through Web-protocols such as REST, Web Services, and RSS/Atom (described below). However, many interesting potential data-sources do not (yet) conveniently expose APIs. Mashups that extract content from sites like Wikipedia, TV Guide, and virtually all government and public domain Web sites do so by a technique known as screen scraping. In this context, screen scraping denotes the process by which a tool attempts to extract information from the content provider by attempting to parse the provider's Web pages, which were originally intended for human consumption.

• The mashup site. This is where the mashup is hosted. Interestingly enough, just because this is where the mashup logic resides, it is not necessarily where it is executed. On one hand, mashups can be implemented similarly to traditional Web applications using server-side dynamic content generation technologies like Java servlets, CGI, PHP or ASP.

Alternatively, mashed content can be generated directly within the client's browser through client-side scripting (that is, JavaScript) or applets. This client-side logic is often the combination of code directly embedded in the mashup's Web pages as well as scripting API libraries or applets (furnished by the content providers) referenced by these Web pages. Mashups using this approach can be termed rich internet applications (RIAs), meaning that they are very oriented towards the interactive user-experience. (Rich internet applications are one hallmark of what's now being termed "Web 2.0", the next generation of services available on the World Wide Web.) The benefits of client-side mashing include less overhead on behalf of the mashup server (data can be retrieved directly from the content provider) and a more seamless user-experience (pages can request updates for portions of their content without having to refresh the entire page). The Google Maps API is intended for access through browser-side JavaScript, and is an example of client-side technology.

Often mashups use a combination of both server and client-side logic to achieve their data aggregation. Many mashup applications use data that is supplied directly to them by their user base, making (at least) one of the data sets local. Additionally, performing complex queries on multiple-sourced data (such as "Show me the average purchase price for real estate bought by actors who have co-starred in movies with Kevin Bacon") requires computation that would be infeasible to perform within the client's Web browser.

• The client's Web browser. This is where the application is rendered graphically and where user interaction takes place. As described above, mashups often use client-side logic to assemble and compose the mashed content.

Ajax

There is some dispute over whether the term Ajax is an acronym or not (some would have it represent "Asynchronous JavaScript + XML"). Regardless, Ajax is a Web application model rather than a specific technology. It comprises several technologies focused around the asynchronous loading and presentation of content:

• XHTML and CSS for style presentation

• The Document Object Model (DOM) API exposed by the browser for dynamic display and interaction

• Asynchronous data exchange, typically of XML data

• Browser-side scripting, primarily JavaScript

When used together, the goal of these technologies is to create a smooth, cohesive Web experience for the user by exchanging small amounts of data with the content servers rather than reload and re-render the entire page after some user action. You can construct Ajax engines for mashups from various Ajax toolkits and libraries (such as Sajax or Zimbra), usually implemented in JavaScript. The Google Maps API includes a proprietary Ajax engine, and the effect it has on the user experience is powerful: it behaves like a truly local application in that there are no scrollbars to manipulate or translation arrows that force page reloads

## Web protocols: SOAP and REST