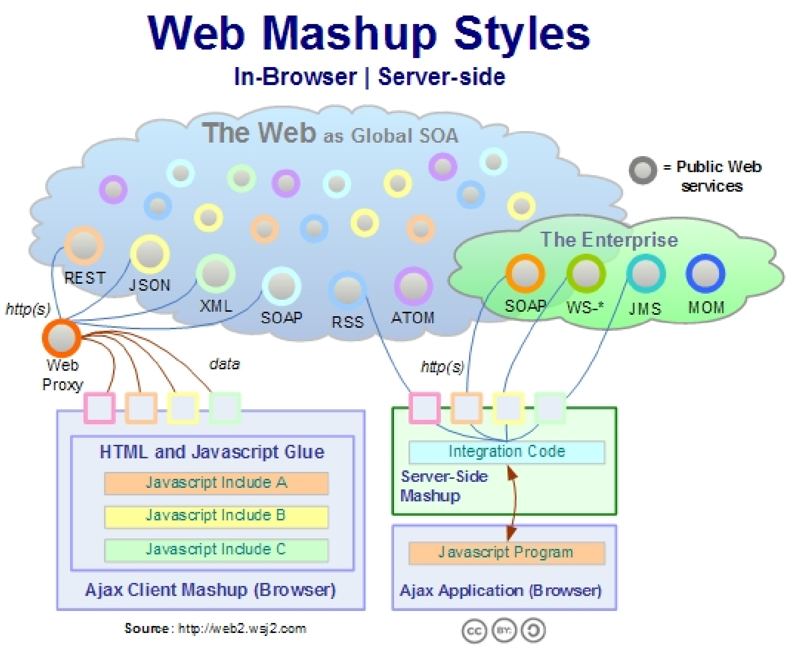
Mashups

The term mashup - as the original term from the music world - describes a Web application that combines multiple external sources (services) into a single application. The external sources are typically other web sites and their data may be obtained by the mashup developer in various ways including, but not limited to: APIs, XML feeds, and screen-scraping. Its like a bit of a buzzword in the Web 2.0 era. It's frequently mentioned in the same context as cloud computing and Web 2.0 while we can access data and tools in the cloud. The primary purpose of most Web mashups is to consolidate information with an easy-to-use interface. Because the combinations of Web applications are limitless, so are the possibilities of mashups. For example, Yahoo offers a mashup called Yahoo! Pipes that aggregates RSS feeds into a single page that can be navigated using a graphical interface. MyWeather.com offers weather information, while Amazon could serve information about new books and Youtube could be used to serve videos and Last.fm to include music recommendation. There are plenty of Mashup tools and editors that really allow users to start playing around with this possibilities. Internet mashup websites are quickly growing for 2007-2008. As of January 2008, there are approximately 12 new internet mashups launched each day. At this time, only a fraction of new mashups achieve significant popularity, but mashups are definitely here to stay. However, not all of the them remained online, Microsoft was closed its visual mashup creator (Popfly) in 2013 - 5 years later from its start. Google also closed its Mashup Editor in 2009, but they provided a solution for the users, they introduced the Google App Engine - as they say: Platform as a Service (PaaS).

An other well-known example from Google is the Maps API, where e.g. a Web forum could display what parts of the world the users are posting from but it can be used to rate areas in a city, delineate points of interest, or show roads that are undergoing construction. Google Maps has spawned thousands of mashup applications. In the previous section we also used it to show an example for the way to Web 3.0 which goes through APIs. This highlights that APIs are key stakeholders currently in the evolution of the Web.

A mashup is a technique by which a website or Web application uses data, presentation or functionality from two or more sources to create a new service. Mashups are made possible via Web services or public APIs that (generally) allow free access. Most mashups are visual and interactive in nature. To a user, a mashup should provide a richer, more interactive experience. To a developer, a mashup is beneficial because it requires less code, allowing for a quicker development cycle.

The overall concept could be seen in the following figure:



Mashups

Characteristics of mashups

We could infer the following outstanding characteristics of mashups, based on the concepts mentioned above:

- A mashup is focused on adding value, putting a tremendous amount of ready information at the fingertips of the final user.
- Information + User experience should be efficiently combined, using technologies such as RIAs.
- It prioritizes the information in the order in which is most likely to be useful. You have to know who you are talking to, what products they have registered, what the top service items are for those products, and then start trying to answer questions that they may have such as where the local service centers are for the customer's location.
- Developing a mashup must be measured in hours or days.
- Generally, it is read-only.
- Mashups are agile views into the data that they present. They are not an all-powerful editing surface capable of editing any and all data thrown at them. If someone wants to edit that data, they should go back to the application that created the data to do the editing. A timestamp may be important for improving the clarity and usefulness of the mashed data.

The 3 categories or the 5 styles of mashups

As we can see these could be combined at multiple levels in an application stack means that there are (at least) five places that mashups can take place. These five styles are:

- Consumer or Presentation Mashup: Information and layout is retrieved and either remix or just placed next to each other. Combine similar types of media and information from multiple sources into a single representation.
- Data Mashups:
    - Client-Side Data Mashup: Takes information from remote Web services, feeds, or even just plain HTML and combines it with data from another source. New information that didn't exist before can result such as when addresses are geocoded and display on a

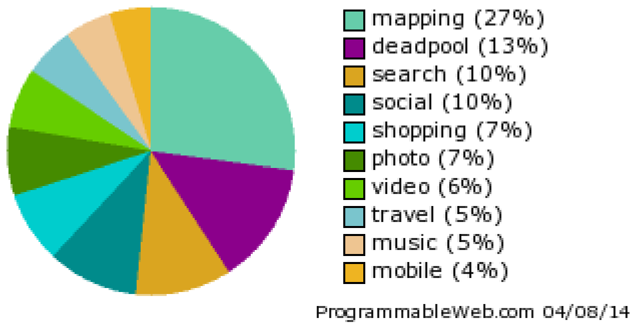
map to create a visualization that could exist without the underlying combination of data.

- Server-Side Data Mashup: Databases have been linking and connecting data for decades, and as such, they have relatively powerful mechanisms to join or mashup data under the covers, on the server-side. While it's still harder to mashup up data across databases from different vendors, products like Microsoft SQL Server increasingly make it much easier to do. This points out that many applications we have today are early forms of mashups, despite the term. Of course, the more interesting and newer aspects of mashups happen above this level.

- Business Mashups:
    - Client-Side Software Mashup: This is where code is integrated in the browser to result in a distinct new capability. While a component model for the browser is only now being hashed out, there is considerable potential in being able to easily wire together pieces of browser-based software into brand new functionality.
    - Server-Side Software Mashup: A better place for integrating software since Web services can more easily use other Web services and there are less security restrictions and cross domain issues. As a result, server-side mashups like those that in turn use things like Amazon's Mechanical Turk or any of the hundreds of open Web APIs currently available, are quite common.

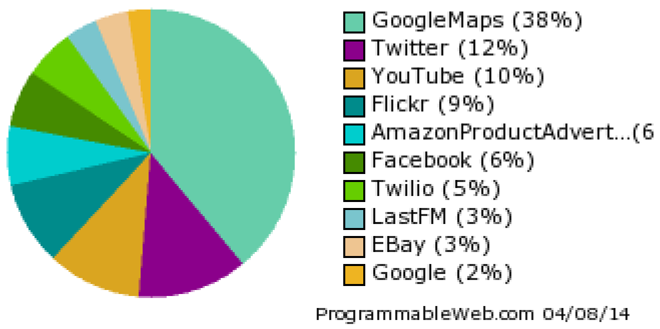
Mashup uses are expanding in the business environment. Business mashups are useful for integrating business and data services, as business mashups technologies provide the ability to develop new integrated services quickly, to combine internal services with external or personalized information, and to make these services tangible to the business user through user-friendly Web browser interfaces. Business mashups differ from consumer mashups in the level of integration with business computing environments, security and access control features, governance, and the sophistication of the programming tools (mashup editors) used. Another difference between business mashups and consumer mashups is a growing trend of using business mashups in commercial software as a service (SaaS) offering. Many of the providers of business mashups technologies have added SOA features.

The presence of these mashups can be easily followed by the http://www.programmableweb.com website which collects all the known mashups and APIs into one central place. It is useful to get some statistics as well. The following figure is listing the mostly used tags for mashups:



Top Mashup categories

followed by the most used APIs:



Top Mashup APIs

Everybody may have a feeling what are the most common usage of the mashups and APIs. All of them try to add some new information to the user but finding the relevant ones are not an easy task. The website shows around 7500 distinct mashups and the average Mashups/Day is 1.5 at the time of this writing. Hard to follow these sites but we can found interesting ones, e.g. one of the highlighted mashup was the Audience Finder. The Audience Finder tool lets you find Facebook users that can be part of a custom audience for your social media marketing campaigns. This way, you can create an ad campaign that is very targeted towards people who are, right now, likely to be interested in your product or service. This is only one usage scenario where mashups and APIs could help us.

Architectural aspects of mashups

Some places refer to mashups as **web applications that combines data from more than one source, hiding this behind a simple unified graphical interface**. Despite for me this is true, I prefer another definition which widens the scope, referring to a mashup as a technique, as an **architectural style for building applications that combine data from multiple sources to create an integrated and improved experience to the user**.

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 product's functionality can be accessed using 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 **façade pattern**. That 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.

The need for the mix: (REST + Traditional SOA) * Web 2.0

In summary, we need the mix before Mashups really take off. I believe both REST and 'Traditional SOA' will work together alongside all the other various ways of getting content on the web to achieve this. Traditional SOA Web Services + RESTful oriented architecture Web Services + RSS Feeds + Social Context + Screen Scraping = Future Mashups.

Prev Up Next
Chapter 3. Web Applications and Mashups Home Mashups versus Portals