The first of a series, this article provides a general overview of the characteristics and technologies related to the term Web 2.0 so that a platform can be laid for a detailed discussion about how they relate to Service-Oriented Architecture (SOA) development. The second part in the series examines the current state of IT and SOA in the enterprise and discusses what situational applications and a mashup ecosystem can offer. The third part describes the IBM® Mashup Starter Kit (IBMMSK) and how you can use it to develop situational applications.

Introduction

Web 2.0 is an extremely broad topic, and for the sake of simplicity, this article serves to synthesize its various components to set a platform for a more detailed discussion around SOA, situational applications, and the IBMMSK in Part 2 of this series. For a broader, more detailed description of Web 2.0, please see link to the article by Tim O'Reilly provided in the Resources section at the end of this article.

What is Web 2.0?

Web 2.0 is best described as a core set of patterns that are observable in applications that share the Web 2.0 label. These patterns are services, simplicity, and community (see a representation of this in Figure 1). After covering the patterns, this article explores some of the technologies that enable them.

Figure 1. Web 2.0 Applications exhibit these three patterns

<table>
<thead>
<tr>
<th>Web 2.0</th>
<th>Services</th>
<th>Simplicity</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Services
To truly empower the Web community, which is an underlying tenet of this new phase of Web usage and application development, the community first has to be able to share a common access method to a given application. To facilitate this, applications are moving to a service model whereby a single instance is accessible by all users. This is opposed to each user having to download and install the application separately. In addition, the shared service model provides the single sandbox that’s required for the community mechanisms to be effective.

Simplicity

Simplicity for the developer: Software is becoming easier to use and develop for certain kinds of applications due to open industry standards and the low barrier to entry of many powerful technologies. This, in turn, is increasing the pool of effective developers and available Web applications. In terms of creating a community, this facet alone is proving to be hugely powerful.

Simplicity for the user: You might also notice that Web applications are now providing rich content and user experiences, and with the advent of Asynchronous JavaScript + XML (Ajax), the line between the traditional stand-alone or thick client and the Web application is now blurred or has disappeared altogether. Current Web applications are going beyond the reload the page to see new content metaphor of Web 1.0 to deliver rich, interactive user experiences.

Because of the success of feed-related technologies (described later in the article), users can now subscribe to syndicated content that provides the specific information they want via industry standard technologies and protocols. Feeds now allow information to be packaged and delivered in a specific format, which allows the information to be easily consumed and remixed. This meets the needs of a trend where software is getting out of the way of information so that information can attain its optimal value (hence a focus on information as opposed to function). This is resulting in service owners seeing the information they provide being used productively in ways they didn’t anticipate.

Community mechanisms
The Web has created a societal shift in the way we organize, access, and use information. One example of this is Wikipedia (see Resources for a link), which enables humanity to work together to collaboratively capture and manage information at a rate that far exceeds a single body’s ability to do the same. This is both good and bad in that it allows information to be added quickly, but the amount of information available is enormous. This leads to issues finding information you need, thus additional community mechanisms are employed to help filter this information so users can now tag information with meaningful and searchable metadata. One example of this is del.icio.us (see Resources for a link), which allows users to store and tag their bookmarks online. This allows other users to search the list of bookmarks by tag, which provides a means of finding a result within data that has already been bookmarked (filtered) by the community.

Integrity of community data: Another pressing issue is the need to vet the integrity of the data the community provides. This same community can be employed to place a value on the information. Users can add and edit comments and recommendations that can be aggregated to provide the community’s collective view on a given resource, thereby truly creating a read and
write Web. This model is also being used to create a tight feedback loop between providers and consumers; an example of this is the buyer and seller rating systems on eBay (see Resources for a link). You can leverage the tight feedback loop of the community to affect software development processes. Software can now be developed in partnership with the user community and, thus, be provided as a continually updated service that improves as adoption increases.

**Connecting people:** Social networking features are further utilized to locate and manage resources of a different kind, namely people. Social networks are a powerful way to locate people around a shared common topic. For example, these networks let people that attended a particular high school or people that belong to a particular industry consortium share, network, and even collaborate in their own "space" on the Web. In addition, these networks can be fairly autonomous in that users in the network can add and invite other users to the network, thereby increasing its value. This is quite different from Web 1.0, which had a focus on connecting computers rather than connecting people.

**The Long Tail and looser license agreements:** Other notable aspects about Web 2.0 applications are that they are more focused on servicing The Long Tail (see Resources for more information) by using technology to reach large numbers of individuals and providing highly personalized products or services versus servicing the masses. In addition, because of Web 1.0's focus on downloadable applications, the license agreements were quite strict (that is, "All rights reserved") in regards to how the applications could be used; with Web 2.0, they generally encourage integration with other technologies, content, and applications (that is, "Some rights reserved").

We see these patterns embodied within popular Web 2.0 flagship Web sites, such as Digg, which employs community mechanisms to add, sort, and comment on information; YouTube, which lets users tag media in a way that's meaningful to them and easily searchable by others; and Facebook, which lets users build and interact within social networks. (See Resources for links to each of these sites.)

**What technologies facilitate the patterns and concepts of Web 2.0?**

This shift to a service-based model has implications for how Web 2.0 applications are now developed. The Web infrastructure is now seen as the bottom of the application development stack (as opposed to a server for a downloadable application). The *programmable Web* is then leveraged as the APIs that you use to build a Web 2.0 application. Examples of these APIs are publicly accessible Web services, such as Flickr, Google Maps, and Digg.

The Linux®, Apache, MySQL, and PHP (LAMP) application development stack is a popular, free, quick, and easy approach to creating Web applications. This has helped lower the on-ramp to creating useful Web applications, which in turn has created a wider base of Web developers. The LAMP stack is the current generation of the technology that has always run a vast portion of the Internet.

The concept that the Web is your new platform is based on the premise that a large part of the infrastructure has already been established and currently exists on the Web. New Web
2.0 applications simply need to leverage existing available hosting and, instead of having to use the traditional layers of APIs—such as Java™ Platform, Enterprise Edition (Java EE) or Microsoft® .NET—you can now build applications using Web-based APIs (or Web services) that are exposed by existing Web applications, such as Flickr and Google Maps. The prevalence of these Web APIs lets you avoid the work of creating certain features, thereby reducing your workload so you can build applications faster.

In addition, you can integrate two or more of these Web APIs to create something new and unique, known as a *mashup*. An example of this is how www.housingmaps.com overlays real estate information (provided by Craigslist) on a map (provided by Google Maps) to allow you to visually navigate and filter real estate information for a given location.

Figures 2 and 3 break down the infrastructure of Web 2.0.

**Figure 2. Key technologies in the Web 2.0 stack**

![Web 2.0 stack diagram](image)

**Figure 3. The traditional application development stack contrasted with the programmable Web**

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Programmable Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Web 2.0 applications</td>
</tr>
<tr>
<td>APIs</td>
<td>Web services</td>
</tr>
<tr>
<td>Operating system</td>
<td>Internet hosting</td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
</tr>
</tbody>
</table>

These Web-based APIs can then be invoked using technologies, such as Ajax, which provides a means for the browser client to communicate with the server via JavaScript (both synchronously and asynchronously). This means the application doesn't require the entire page to be reloaded every time the client needs to communicate with the server.

You can use JavaScript Object Notation (JSON) to serialize and deserialize objects so that they can be sent between the browser client and the server via Ajax. Much like Ajax, the SOAP specification and the Representational State Transfer (REST) approach can also be used to...
provide an industry standard means of transferring data between applications. It's now quite common to see existing services provide SOAP, Ajax, and REST interfaces.

REST is a style for designing services using simple HTTP actions working on URL-addressable entities.

Additional technologies, such as Dojo—which is a time-saving, open source, cross-browser Dynamic HTML (DHTML) toolkit—and browser extensions, such as Greasemonkey and Firebug—which let you write scripts that extend the browser and tailor the way a Web page looks or functions—can leverage Ajax and provide for the rich user interfaces available today. For instance, you can install a Greasemonkey script for Facebook so that whenever you move the mouse pointer over a thumbnail of a picture, the full-size picture automatically appears in the left side of the browser window. This makes viewing images both quick and convenient, and definitely enriches the user experience.

Microformats let you use XHTML to convey semantic meaning within an (X)HTML, Atom, RSS, or arbitrary XML document. The microformats are machine and human readable-allowing applications, such as browsers to autodiscover them and act on the information they describe. For example, if you want to embed a business card in a document, you can use the hcard microformat (see Resources for more information), which uses XHTML to describe your business card (see Figure 4).

Figure 4. An example of the hcard microformat

```html
<div class="vcard">
  <div class="fn org">Wikimedia Foundation Inc.</div>
  <div class="adr">
    <div class="street-address">200 2nd Ave. South #358</div>
    <span class="locality">St. Petersburg</span>,
    <abbr class="region" title="Florida">FL</abbr>
    <span class="postal-code">33701-4313</span>
  </div>
  <div class="tel">+1-727-251-0101</div>
  <div class="email">info@wikimedia.org</div>
  <div class="fax">+1-727-256-0207</div>
</div>
```

Lastly, RSS and Atom feeds have now gone beyond being used only to subscribe to blogs and news feeds, and are seen as potential approaches to simplify specific content-centric application architectures. The Atom specification is a more recent evolution of the ideas originally embodied by RSS and provides useful features, such as the Atom Publishing Protocol (APP), which lets you publish information to be added to a feed. If event-driven architectures are now part of the SOA framework, then feeds can be considered part of the service paradigm and should be leveraged as such.
Conclusion

In this article, you learned about core patterns, concepts, and key technologies that are embodied in Web 2.0 applications. In the follow-up article, Part 2, you'll take a look at IT in the enterprise and investigate how a mashup ecosystem lets you build situational applications that improve on the current state of the art.

Acknowledgements

My thanks go out to Stew Nickolas, Dan Gisolfi, and Rick Robinson for reviewing and contributing to this article.

© Copyright IBM Corporation 2007
Trademark
(www.ibm.com/developerworks/ibm/trademarks/)