CHAPTER 2

Uncovering the Mashup Potential of Web Sites

In the previous chapter, you studied several examples of mashups in depth. With the goal of learning how to create your own mashups, you'll turn now to the raw ingredients of mashups—individual web sites and web applications. Although the focus of this book is on public web application programming interfaces (APIs), you'll first study the human user interface (UI) of web sites for their mashup potential.

Why not jump straight to using APIs if that's what you want to use to create mashups? After all, wouldn't the APIs be the most useful place to begin with since they are especially designed for enabling access to the web site's data and services? What you learn from studying a web site's user interface is useful—even essential—to using APIs effectively. When you exercise a web site's public API, you usually need to understand the overall logic of the web site. For instance, some mashups, such as those created with Greasemonkey (like the Google Maps in Flickr [GMiF] script from Chapter 1), extend the application directly by hooking into and blending with the existing user interface. To create something like GMiF, you would need detailed knowledge of the application you plan to mash up. One of the best ways to uncover potential hooks of a web site is to use the web site as an end user, armed with a developer's sensibility.

Creating mashups doesn't always require much programming. It can be as simple as linking to the right part of an application, accessing the appropriate feed, or connecting the web site to a weblog. In this chapter, I will point out how features created for end users can enable you to create mashups with minimal or no programming.

Flickr is the central example in this chapter, one that I analyze extensively. I follow with Google Maps as an important complementary example. Flickr and Google Maps are among the most mashed up APIs on the Web. I also discuss del.icio.us, a pioneering social bookmarking site, and Amazon, which is an example of an e-commerce platform. In this chapter, I have selected highly remixable applications—as opposed to web sites that are difficult to recombine—as a way to ease into your study of creating mashups.

In this book, I focus mostly on how to use public APIs but briefly mention screen-scraping. APIs often don't do everything you might want from them. Although you can do a lot with public APIs, screen-scraping provides an important alternative or complementary approach. Nonetheless, you should use the API as the first resort. You can screen-scrape if you need to, but always use a web site's computational and network resources respectfully, being mindful of the legal ramifications of what you are doing.

What Makes Web Sites and Applications Mashable

I'll now cover the aspects of web sites and web applications that make them amenable to mashups. Some features are useful regardless of whether you are using the API or whether you are using informal mechanisms for integration. In either case, you are looking for ways to hook into an application. The following sections will help you to analyze a web site for these integration hooks.

UNIDERSTANDING THE TERMINOLOGY

You want to be careful to distinguish different uses of the term *hacking*. When I say you want a site to be *hackable*, I don't mean people should easily be able to break the security elements of the web site. Such activity isn't hacking—that's *cracking*. When you design a site to be hackable, you are designing it to be extensible, even in what you hope to be highly transformative ways. See Eric Raymond's Jargon File web site for a relevant definition of a hacker (`http://www.catb.org/jargon/html/H/hacker.html`), especially the following:

* "A person who enjoys exploring the details of programmable systems and how to stretch their capabilities"
* "One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations"

Note the deprecated usage: "A malicious meddler who tries to discover sensitive information by poking around. Hence *password hacker*, *network hacker*. The correct term for this sense is *cracker*."

Some people are talking about "designing for hackability" (`http://www.brianoberkirch.com/2007/04/03/designing-for-hackability/`).

I'm using the term *reverse engineering* to refer to a careful study of a web site, its functionality, and how it's put together. I outline some techniques, but there are more to use. Reverse engineering is a long- honored tradition in this society—but you need to be aware of some of the legal and ethical issues of it. Please refer to `http://www.chillingeffects.org/reverse/faq.cgi` for some information. (What I write here, of course, is not legal advice on reverse engineering.)

Ascertaining the Fundamental Entities of the Web Site

The basic questions to begin with when analyzing a web site are the following: What is the web site fundamentally about? What are the key entities, or *resources* to borrow a term from W3C parlance? How are these entities or resources associated with specific URLs/URIs? A resource is anything with a URI associated with it. A formal definition of a resource comes from "Uniform Resource Identifier (URI): Generic Syntax" (RFC 3986):[1]

> *This specification does not limit the scope of what might be a resource; rather, the term "resource" is used in a general sense for whatever might be identified by a URI. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles"), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound*

books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity).

1. http://tools.ietf.org/html/rfc3986#section-1.1

The question of resources and their corresponding URIs are not as abstract as they may sound. In fact, looking at resources may seem rather obvious. For example, for Flickr, which is self-described as "almost certainly the best online photo management and sharing application in the world," important entities are, not surprisingly, photos and users. As you will see later in the chapter, these entities are also resources; you can identify specific photos and users in the URLs produced by Flickr. For example, this URL:

```
http://www.flickr.com/photos/raymondyee/508341822/
```

is for photo `508341822`, which belongs to user `raymondyee`. A Flickr photo is *addressable* via a URL; that is, a URL can lead you right to the photo in question. As experienced users of the Web, we all know the useful things we can do when there are specific URLs. You can bookmark a link, e-mail it, and use it as a reference in a web page. You don't have to tell someone to go to Flickr and type in the photo number to get to the photo.

As you will see later in this chapter, granular URLs also enable mashups. A major part of this chapter is devoted to studying web sites by analyzing their end-user functionality and how it can be seen through its URI structure (*URL language*). In the following sections, I discuss in greater detail notions of addressability, granularity, transparency, and persistence in URLs. I will present a detailed listing of entities and how you can refer to them in URIs for Flickr, as well as a brief analysis of Google Maps, Amazon, and del.icio.us for comparison.

Public APIs and Existing Mashups

Is there a public API for the web site? A web site's public API is specifically designed as the official channel of programmatic access to the data and services of the web site. It essentially lets you access and program the web site almost like a local object or database. For a slightly more formal definition of an API, consider the one by John Musser from Programmableweb.com: "a set of functions that one computer program makes available to other programs so they can talk to it directly."[2] Although there are APIs for operating system, applications, and programming toolkits, this book focuses on the APIs of web sites and web applications.

If there is a public API for a web site, how have people used the API? Looking for what others have done with the API helps you get right into the application without wading through any documentation.

What's the range of the third-party wrappers available for the API? How many are officially supported by the web site owners? Which ones were developed by the community?

Are there many people working with the API, or is there little evidence that it is being used at all? Have any mashups using the web site been developed? How sophisticated are the mashups? Are they straight-up remixes of the data or presentations of the data in a new context? Do you see some emergent and unexpected property?

2. http://programmableweb.com/faq#Q2

Surprising mashups often reveal a capacity in the formal API or some integration point that might not be obvious from a quick glance at the documentation.

The more interesting mashups that exist for an application, the more likely it is that the application is amenable to mashups. Look for mashups that contain functionality similar to what you want to include.

Chapter 6 and Chapter 7 present an overview of how to use public web site APIs, starting with a study of the Flickr APIs and then moving on to a survey of other APIs.

Use of Ajax

Does the web site use Ajax and allied JavaScript techniques to integrate data dynamically into the user interface? As you will learn in Chapter 8, the presence of Ajax is an indicator that there is likely an API at work—either a formalized public API or a programmatic structure, though not intended for public interfacing, that might possibly be used for mashup making. (Recall from Chapter 1 how Housingmaps.com placed markers on the first generations of Google Maps by tapping into the programming logic before any public API for Google Maps was released.)

Embedded Scriptability

Can people embed plug-ins, add-ons, or extensions (as opposed to writing external applications) to extend the web site directly? Here are examples of extension frameworks for specific web sites:

* Google Gadgets (`http://www.google.com/ig/directory`) to extend iGoogle (`http://www.google.com/ig`) and Google Mapplets (`http://www.google.com/apis/maps/documentation/mapplets/`) to extend Google Maps (`http://maps.google.com/maps/mm?mapprev=1`)
* Microsoft Web Gadgets (`http://gallery.live.com/`) to extend Windows Live (`http://live.com`)

For web applications, these are some examples:

* WordPress plug-ins (`http://codex.wordpress.org/Plugins`)
* MediaWiki extensions (`http://www.mediawiki.org/wiki/MediaWiki_extensions`)

For desktop applications/OS environments, take a look at these examples:

* Microsoft Office macros and add-ins (`http://msdn2.microsoft.com/en-us/office/default.aspx`)
* OpenOffice.org macros, add-ins, and add-ons (`http://wiki.services.openoffice.org/wiki/Extensions`)
* Yahoo! Widgets (`http://widgets.yahoo.com/`)
* SketchUp Ruby (`http://www.sketchup.com/?sid=79`)

If you have the required permissions, you can install or write extensions that incorporate other services into the applications.

Browser Plug-Ins

Are there Firefox add-ons (https://addons.mozilla.org/en-US/firefox/) that supplement or enhance the user interface to the web site? For example:

* Better Flickr Firefox Extension (http://lifehacker.com/software/lifehacker-code/upgrade-flickr-with-the-better-flickr-firefox-extension-263985.php)
* The del.icio.us Firefox extension (http://del.icio.us/help/firefox/extension)
* S3Fox Amazon S3 Firefox Organizer (https://addons.mozilla.org/en-US/firefox/addon/3247)

If you see a form of communication between the add-on and the application, you know there is some form of public or private API. Other browsers have extension mechanisms,[3] but I single out Firefox add-ons because you can unzip the add-on to study the code (if it hasn't been obfuscated) to gain more insight into the hooks of the corresponding web site.

Getting Data In and Out of the Web Site

How can you import data into the application? With what protocols? What data or file formats are accepted?

How can you export data from the application? What formats are provided? What protocols are supported?

It's much easier to make mashups out of widely deployed data formats and protocols (whether they are *de jure* or *de facto* standards) than with obscure data formats and protocols.

Can you embed data from the web site elsewhere? An example of such embedding is a JavaScript badge (such as http://www.platial.com/mapkit/faq). What options do you have for customizing the badge? Super-flexible badges can be used themselves to access data for mashups and hint at the existence of a feature-rich API.

The Community of Users and Developers

What communities of users and developers have grown around the web site? Where can you go to participate in that community and ask questions? What are members of the community discussing? What are some of the limitations of the application that they want to be overcome? What clever solutions or workarounds—*hacks*—are being popularized in that community, not only among developers but also among nonprogramming power users in the community?

Again, seeing how the API gets used and discussed is a great way to get a handle on what is possible and interesting. And if you don't see much activity around the API, realize that you are likely to be on your own if you decide to use it.

Why do I stress looking at the community around an application and its API? A vibrant and active community makes a lot of mashup work practical. When making mashups, some things are theoretically possible to do—if you had the time, energy, and

resources—but are practically impossible for you as an individual to pull off. A community of developers means there are other people to work with, lots of examples of what other people have done, and often code libraries that you can build upon.

3. http://blog.mashupguide.net/2007/04/29/browser-extension-mechanisms-for-various-browsers/

Mobile and Alternative Interfaces and the Skinnability of the Web Site

How many versions of the user interface are there for the web site? Is there a mobile interface? A mobile version is often easier to decipher than the main site and highlights what the web site's creators believe to be some core logic of the web site. A mobile version might be more easily integrated into a mashup for a phone; there is typically no JavaScript to worry about, and the HTML is easier to parse.

How difficult is it to change the look of the interface? That is, how "skinnable" is the web site? Easy customizability of the interface for end users is an indicator that the application developers have likely separated the application logic from presentation logic. And if skinnability is available to end users, that functionality might also be programmable. For example, WordPress themes typically allow the owner of a WordPress site to change the set of global styles of the site.

Documentation

Good documentation of the features, the API, the data formats, and any other aspect of the web site makes it much easier to understand and recombine its data and functionality. Are the input and output data documented? If so, are there schemas, in other words, ways to validate data? Are the formats properly versioned?

Documentation reduces the amount of guesswork involved. Moreover, it brings certainty to whether a function you uncover through reverse engineering is an official feature or an undocumented hack that has no guarantee of working for any length of time.

Is the Web Site Run on Open Source?

If the web site is powered by free or open source software, you have the option of studying the source directly should reverse engineering—or reading the relevant documentation—not give you the answers you need.

Intellectual Property, Reusability, and Creative Commons

Does the web site allow users to explicitly set the licensing of content and data, under Creative Commons, for instance? Does the web site enable users to search and browse content by license? Explicit licensing of digital content clears away important barriers to creating mashups with that content. A detailed discussion of the Creative Commons is beyond the scope of this book. To learn more, consult the following:

`http://creativecommons.org`

Tagging, Feeds, and Weblogging

Here I present a series of questions that will be explored at length in the chapters that immediately follow.

Does the web site use tagging? That is, can users tag items and search for items by tags in the web site? Chapter 3 covers tagging and folksonomy in detail and shows how tags provide mashups with hooks within a web site and among web sites.

Are there RSS and Atom feeds available from the site? Do they give you fine-grained access to the web site? (That is, can you get feeds for a specific search term or for a specific part of a web site?) In the absence of a formal API, syndication feeds become a source of structured, easy-to-parse data. See Chapter 4 for detailed coverage of RSS and Atom feeds.

Does the web site allow you to send content to a weblog or wiki? Studying how the web site is connected to a weblog in this manner is an excellent way to get some practice with configuring APIs without programming. See Chapter 5 for more on blogging and wiki APIs.

URL Languages of Web Sites

I will spend most of this chapter analyzing a web site's functionality by explaining the way its URLs relate to its various entities and resources. In other words, I decipher the web site's *URL language*. At the beginning of the chapter, I already made an argument for the usefulness of having URLs that give you direct access to a resource. Before analyzing Flickr, Google Maps, Amazon, and del.icio.us for their URL languages, I'll make some general comments about URL languages. Each web site has its own URL language, but URL languages vary in terms of addressability, granularity, transparency, and persistence.[4]

4. Restful Web Services by Leonard Richardson and Sam Ruby (O'Reilly Media, 2007). The idea of analyzing URIs in terms of addressability, granularity, transparency, and persistence comes from Restful Web Services.

Leonard Richardson and Sam Ruby present a helpful definition of *addressability*: "Addressability means that every interesting aspect of your service is immediately accessible from outside. Every interesting aspect of your service has a URI: a unique identifier in a format that's familiar to every computer-literate person.... Addressability makes it possible for others to make mashups of your service: to use it in ways you never imagined."

Some URL languages are highly expressive, making resources and their associated data addressable at high *granularity*. Others expose relatively little of the functionality of the web site or only at a very course-grained level. Some URL languages are relatively *transparent*; their meaning and context are easily apparent to those who did not design the site. Other URL languages tend to the opaque, making it difficult or impossible to refer to web site's functionality in any detail. Finally, some URL languages have URLs that have high *persistence*, which means to last, while others do not, making them difficult to link to.

UNDERSTANDING THE RELATIONSHIP AMONG URI, URL, AND URN

Universal Resource Identifier (URI) is a specific type of identifier. URIs fall into two classes: Universal Resource Locators (URLs) and Universal Resource Names (URNs). You're likely to be much more

familiar with the former. An example of the latter is urn:isbn:159059858X, which refers to this book. Many of the things I write about URLs in this book apply to URIs in general.

RFC 3986 clarifies the relationship among URI, URL, and URN (http://tools.ietf.org/html/rfc3986#section-1.1.3):

> *A URI can be further classified as a locator, a name, or both. The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). The term "Uniform Resource Name" (URN) has been used historically to refer to both URIs under the "urn" scheme [RFC2141], which are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable, and to any other URI with the properties of a name.*
>
> *An individual scheme does not have to be classified as being just one of "name" or "locator." Instances of URIs from any given scheme may have the characteristics of names or locators or both, often depending on the persistence and care in the assignment of identifiers by the naming authority, rather than on any quality of the scheme. Future specifications and related documentation should use the general term "URI" rather than the more restrictive terms "URL" and "URN" [RFC3305].*

Some Mashups Briefly Revisited

Let's see how the ideas and questions presented so far in this chapter (for example, studying the user interface of the applications, their URL languages, and how they exploit certain hooks such as RSS) would have come in play for the developers of the mashups in Chapter 1:

* Housingmaps.com depends on a combination of screen-scraping the web pages of Craigslist and the RSS feeds of Craigslist since Craigslist doesn't have a public API. (Chapter 9 is a fuller analysis of the logic behind Housingmaps.com.)
* The construction of Google Maps in Flickr (GMiF), a Greasemonkey script that sticks an icon into the Flickr interface, rests on understanding the UI of Flickr, how the user community geotags photos, and how to screen-scrape information that is in the HTML page—in addition to the public API of Flickr.
* Creating the LibraryLookup bookmarklet depended centrally on speaking the URL languages of both the source of book information (that is, Amazon) and the destination for the query (that is, library catalogs).

Flickr: The Fundamentally Mashup-Friendly Site

Let's start our study of highly remixable web sites with Flickr, looking for features that make this site amenable to mashups. In addition to many features for storing and sharing photos, Flickr is chock-full of features that make it easy to mash up Flickr, not least of which is its use of XML, XML web services, tagging, and Ajax. These features are

blended in a surprisingly coherent, comprehensive demonstration of how to knit this new technology together. You won't be surprised then to read Flickr's own description of its goals at `http://www.flickr.com/about`:

> *To do this, we want to get photos into and out of the system in as many ways as we can: from the web, from mobile devices, from the users' home computers and from whatever software they are using to manage their photos. And we want to be able to push them out in as many ways as possible: on the Flickr web site, in RSS feeds, by e-mail, by posting to outside blogs or ways we haven't thought of yet. What else are we going to use those smart refrigerators for?*

This flexibility of functionality makes Flickr a good web site to study when learning about mashups. Flickr is made to be mashed up. You can see in one web site a great variety of mashup-enabling techniques—and actually how they can be well integrated in one web site.

Caution In the following analysis of Flickr, I answer many but not necessarily all the questions listed earlier in the chapter. I'll answer some of the questions later in the book. Moreover, it's important to understand that since Flickr is a constantly evolving web site, any of the details recorded here about Flickr can become out of date. My hope is to provide you with enough ways to think about web sites so that you will be able to adapt to those changes.

Resources in Flickr

Let me now explain Flickr in terms of its URI language, detailing resources and their corresponding URIs. As you use Flickr as an end user and think about its functionality in terms of key entities, you will probably come up with a list similar to the following:

* Users (or people)
* Notes
* Photos
* Comments (for both photosets and photos)
* Tags
* Licenses
* Archives
* Prefs
* Sets (which are also called *photosets*)
* Groups
* Collections
* Contacts

* Favorites
* Blogs
* Geo

This list is meant to cover the broad range of what Flickr does, but I'm not attempting to be exhaustive. Remember that there are different ways to slice the pie, so any listing of resources won't necessarily agree. We will end up agreeing on how the URLs are structured, though.

How did I come up with this list?

* I used Flickr, looking at each piece of functionality available to me. For each function, I identified the "nouns," or entities, at work and noted the corresponding URIs and how the URLs change as the state of the application changes.
* I culled common terminology from the Flickr UI itself, from the documentation of the UI, and from the documentation for the API (`http://www.flickr.com/services/api/`). The structure of an API often points out key entities in the web site.

Caution Keep in mind the warning about the opacity of unique identifiers in Flickr: "The Flickr API exposes identifiers for users, photos, photosets and other uniquely identifiable objects. These IDs should always be treated as opaque strings, rather than integers of any specific type. The format of the IDs can change over time, so relying on the current format may cause you problems in the future."[5]

---

5. http://www.flickr.com/services/api/misc.overview.html

Users and Photos

The host URL of the entire site is as follows:

```
http://www.flickr.com/
```

URLs using a host URL of `http://flickr.com` also seem to be valid, but I will use the former since the API returns URLs that use www.flickr.com as the host URL.

Since Flickr is a social photo-sharing site, let's start with the two entities you expect at the least: a Flickr user (or person) and a photo.

Note I use URI Templates (`http://bitworking.org/news/URI_Templates`) to express the URL language. These are strings into which I place variables that are replaced to form the full URI. The variables to be substituted are delimited by `{}` (which are not part of legal URIs). Note that the URI Template is currently an IETF draft, but the convention I use here is simply denoting the embedded variable with `{}`. Substituted variables need to be properly URL encoded (`http://en.wikipedia.org/wiki/Percent-encoding`).

The profile page for a user, the URL that most closely represents a Flickr user, is as follows:

```
http://www.flickr.com/people/{user-id}/
```

The `user-id` can take one of two forms:

* An NSID (a unique identifier that contains a `@` character) generated by Flickr when the user signs up for an account (for example, `48600101146@N01`)
* A custom URL handle or "permanent alias" chosen by the user, which can be set at `http://www.flickr.com/profile_url.gne` (for example, `raymondyee`)

My profile page is thus accessible as either this:

```
http://www.flickr.com/people/48600101146@N01/
```

or this:

```
http://www.flickr.com/people/raymondyee/
```

As a logged-in user, you can upload photos to your account using the following form:

```
http://www.flickr.com/photos/upload/
```

Photos belonging to a user are collected here:

```
http://www.flickr.com/photos/{user-id}/
```

Representations of a Photo

Every photo belongs to one specific user, has a unique identifier `photo-id`, and is associated with a URL:

```
http://www.flickr.com/photos/{user-id}/{photo-id}/
```

For example:

```
http://www.flickr.com/photos/raymondyee/508341822/
```

A given photo has a variety of representations, as documented here:

```
http://www.flickr.com/services/api/misc.urls.html
```

When you upload a photo to Flickr, Flickr retains the original image and generates versions (in different sizes) of the photo, as recorded in Table 2-1.

Table 2-1. Representations of a Flickr Photo

photo-type	context-type	Image Type	Sizes of Photo
`s`	`sq`	Small square	75~TMS75
`t`	`t`	Thumbnail	100 on longest side
`m`	`s`	Small	240 on longest side
blank	`m`	Medium	500 on longest side
`b`	`l`	Large	1024 on longest side

o	o	Original image, either a JPG, GIF, or PNG, depending on source format

There are two types of URLs for each size of photo:

* The context page for the photos
* The photos themselves in their various sizes

The context page is of the following form:

```
http://www.flickr.com/photo_zoom.gne?id={photo-id}&size={context-type}
```

where `context-type` is one of `sq`, `t`, `s`, `m`, `l`, or `o`. Not every `context-type` is available for any given photo. (Some photos are too small; nonpaying Flickr members cannot offer original photos for downloading.)

To understand the URLs for the photos themselves, you need to know that in addition to `photo-id` for every photo, there are the following parameters:

* `farm-id`
* `server-id`
* `photo-secret`
* `original-secret`
* `file-suffix`

The URL for the photos takes one of three slightly different forms:

* For the original photo, it is as follows where `file-suffix` is `jpg`, `gif`, or `png`:

```
http://farm{farm-id}.static.flickr.com/{server-id}/{photo-id}_{o-secret}_o.
{file-suffix}
```

* For all the derived sizes except the medium size, the URL is as follows:

```
http://farm{farm-id}.static.flickr.com/{server-id}/{photo-id}_{photo-secret}_

{photo-size}.jpg
```

* For medium images, the URL is as follows:

```
http://farm{farm-id}.static.flickr.com/{server-id}/{photo-id}_{photo-secret}.jpg
```

Let's consider `http://www.flickr.com/photos/raymondyee/508341822/` as an example. If you go to the URL and hit the All Sizes button, you'll see the various sizes that are publicly available for the photo. If you click all the different sizes and look at the URLs for the photos and the context pages, you can determine the values listed in Table 2-2, thus confirming the values of the parameters in Table 2-3.

Table 2-2. URLs for the Various Sizes of Flickr Photo 508341822

Image Type	Context Page URL	Image URL
Small square	`http://www.flickr.com/photo_zoom.gne?id=508341822&size=sq`	`http://farm1.static.flickr.com/193/508341822_2f2bfb4796_s.jpg`

Thumbnail	http://www.flickr.com/photo_zoom.gne?id=508341822&size=t http://farm1.static.flickr.com/193/508341822_2f2bfb4796_t.jpg
Small	http://www.flickr.com/photo_zoom. gne?id=508341822&size=s http://farm1.static.flickr.com/193/508341822_2f2bfb4796_m.jpg
Medium	http://www.flickr.com/photo_zoom. gne?id=508341822&size=m http://farm1.static.flickr.com/193/508341822_2f2bfb4796.jpg
Large	http://www.flickr.com/photo_zoom. gne?id=508341822&size=l http://farm1.static.flickr.com/193/508341822_2f2bfb4796_b.jpg
Original	http://www.flickr.com/photo_zoom. gne?id=508341822&size=o http://farm1.static.flickr.com/193/508341822_5ab600db14_o.jpg

Table 2-3. Parameters Associated with Photo 508341822

Parameter	Value
photo-id	508341822
farm-id	1
server-id	193
photo-secret	2f2bfb4796
original-secret	5ab600db14
file-suffix	jpg

Tip I suggest you look at the current documentation for the Flickr URLs every so often because the URLs that Flickr produces have changed over time, and I suspect they will continue to change as Flickr scales up its operations. Don't worry about any URLs you have generated according to older schemes—Flickr tries to keep them working. (It's worthwhile to update your software to use the latest URL structures if you are able to do so.)

Data Associated with an Individual Photo

Each photo has various pieces of information associated with it, including the following:

* Title
* Description
* Tags
* Machine tags
* Dates (the time it was uploaded as well as the time it was taken, if that time is available)
* EXIF data
* Owner of the picture
* Any sets to which the photo belongs

* Any groups to which the photo belongs
* Comments
* Notes
* Its visibility

I listed these data elements associated with each picture because each of the elements is an opportunity for integration if you want to use that picture in another mashup context. Many of data elements can be addressed in the URL, which is part of the Flickr URL language.

Miscellaneous Editing of Attributes

If you have JavaScript turned on in your browser while accessing Flickr, you might not see the distinct URL for editing the tags, description, and title of the photo—beyond the URL for the photo itself:

```
http://flickr.com/photo_edit.gne?id={photo-id}
```

You can see the EXIF data of a photo here:

```
http://www.flickr.com/photo_exif.gne?id={photo-id}
```

For example:

```
http://www.flickr.com/photo_exif.gne?id=688436870
```

You can edit a photo date here:

```
http://www.flickr.com/photo_date_taken.gne?id={photo-id}
```

Tags

Tags are one of the most important ways to organize photos in Flickr. Tags are words or short phrases that the owner (or others with the proper permission) can associate with a photo. A tag typically describes the photo and ties together related photos within a user's collection of photos and sometimes between photos of different users. However, there is no requirement that tags have meaning to anyone except the tagger, or even the tagger! See Chapter 3 for an extended discussion on tagging and folksonomy.

Flickr lets users search and browse photos by tags. First, let's study how to address tags as they are used throughout Flickr to describe pictures among all users. Then, you will examine the functionality in the context of a specific user.

You can see a list of popular tags in Flickr here:

```
http://www.flickr.com/photos/tags/
```

Popular tags allow you to get a sense of the Flickr community, over the longer haul, as well as over the last 24 hours or 7 days.

The URL for the most recent photos associated with a `tag` is as follows:

```
http://www.flickr.com/photos/tags/{tag}/
```

For example:

```
http://www.flickr.com/photos/tags/flower/
```

You can page through the photos with this:

```
http://www.flickr.com/photos/tags/flower/?page={page-number}
```

Instead of sorting photos by the date uploaded, you can see sort them by descending "interestingness" (a quantitative measure calculated by Flickr of how interesting a photo is):

```
http://www.flickr.com/photos/tags/{tag}/interesting/
```

Finally, for some tags, Flickr identifies distinct clusters of photos, which you can access here:

```
http://www.flickr.com/photos/tags/{tag}/clusters/
```

For example:

```
http://www.flickr.com/photos/tags/flower/clusters/
```

You can display the popular tags used by a specific user here:

```
http://www.flickr.com/photos/{user-id}/tags/
```

You can list all the user's tags here:

```
http://www.flickr.com/photos/{user-id}/alltags/
```

You can show all photos with a given tag for a specific user here:

```
http://www.flickr.com/photos/{user-id}/tags/{tag}/
```

You can edit the tag for the given user, if you have permission to do so, here:

```
http://www.flickr.com/photos/{user-id}/tags/{tag}/edit/
```

You can delete a tag here:

```
http://www.flickr.com/photos/{user-id}/tags/{tag}/delete/
```

You can show a slide show of these tagged photos here:

```
http://www.flickr.com/photos/{user-id}/tags/{tag}/show/
```

User's Archive: Browsing Photos by Date

You can browse through a user's photos by date—by either the date the photo was taken or when it was uploaded. Dates are an excellent way to organize resources such as photos. Even if you leave a photo completely untagged, Flickr can at the very least place the photo in the context of other photos that were uploaded around the same time. If you are careful about generating good time stamps for your photos, you can display photos in an accurate time stream. I have found looking at a user's photos by date to be an effective way to make sense of large numbers of photos.

The main page for a user's archive is here:

```
http://www.flickr.com/photos/{user-id}/archives/
```

For example:

```
http://www.flickr.com/photos/raymondyee/archives/
```

You can sort your archive by the date taken or date posted with this:

```
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
```

where {date-taken-or-posted} is date-taken or date-posted.

You can view the photos for a given date with a different {archive-view} here:

```
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{archive-view}
```

where {archive-view} is one of detail, map, or calendar.

You can also set the display option and limit photos by year, year/month, or year/month/date. The following set of URLs use the default list view:

```
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/{year}
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{month}
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{month}/{day}
```

The following URLs use the other display options where {archive-view-except-calendar} is either detail or map—but not calendar:

```
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/{year}
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{archive-view}
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{month}/{archive-view}
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{month}/{day}/{archive-view-except-calendar}
```

Here are some specific examples:

```
http://www.flickr.com/photos/raymondyee/archives/date-taken/2007/
http://www.flickr.com/photos/raymondyee/archives/date-taken/2007/06/22/
http://www.flickr.com/photos/raymondyee/archives/date-posted/2007/calendar/
```

Sets

Sets or *photosets* (both terms are used in the Flickr UI and documentation) are groupings created by users of their own photos. (Note that sets cannot include other users' photos.)

You can see a user's sets here:

```
http://www.flickr.com/photos/{user-id}/sets/
```

You can see a specific set with the unique ID set-id here:

```
http://www.flickr.com/photos/{user-id}/sets/{set-id}/
```

You can control the view for a given set here where set-view is one of detail, comments, or show:

```
http://www.flickr.com/photos/{user-id}/sets/{set-id}/{set-view}
```

Consider some examples of sets:

```
http://www.flickr.com/photos/raymondyee/sets/72157600434284985/
http://www.flickr.com/photos/raymondyee/sets/72157600434284985/detail/
```

To display a photo in the context of a containing set, use this:

`http://www.flickr.com/photos/{user-id}/{photo-id}/in/set-{set-id}/`

For example:

`http://www.flickr.com/photos/raymondyee/591991800/in/set-72157600434284985/`

Collections

Users can create collections to make groupings of their sets. A user's collections are found here:

`http://www.flickr.com/photos/{user-id}/collections/`

And you can find a specific collection here:

`http://www.flickr.com/photos/{user-id}/collections/{collection-id}`

For example:

`http://www.flickr.com/photos/raymondyee/collections/72157600592620295/`

Favorites

Users can add other users' photos to their favorites:

`http://www.flickr.com/photos/{user-id}/favorites/`

Note that you can't add your own photos to your favorites. There are also not many ways to organize your favorites. You can search within your favorites using this:

`http://www.flickr.com/search/?w=faves&q={search-term}`

Since sets and collections can contain only those photos belonging to a user, there is no built-in way in Flickr for you to group your own photos with photos belonging to others.

A User's Popular Photos

Users can track which of their photos are the most popular (by interestingness, number of views, number of times they have been added as a favorite, and number of comments) here:

`http://www.flickr.com/photos/{user-id}/{popular-mode}/`

where `{popular-mode}` is one of `popular-interesting`, `popular-views`, `popular-faves`, or `popular-comments`. Users can access popularity statistics for only their own photos.

Contacts

As a social photo-sharing site, Flickr allows users to maintain a list of contacts. From the perspective of a registered user of Flickr, there are five categories of people in Flickr: the user, the user's family, the user's friends, the user's contacts who are neither family nor friend, and everyone else. Contacts, along with their recent photos, belonging to a user are listed here:

`http://www.flickr.com/people/{user-id}/contacts/`

Depending on access permissions, you may be able to access more fine-grained lists of contacts for a user here where `{contact-type}` is one of `family`, `friends`, `both`, or `contacts`:

```
http://www.flickr.com/people/{user-id}/contacts/?see={contact-type}
```

Users can see their own list of users they are blocking here:

```
http://www.flickr.com/people/{user-id}/contacts/ignore/
```

Users can see their "reverse contacts" (users who consider them contacts) here:

```
http://www.flickr.com/people/{user-id}/contacts/rev/
```

To invite others to join Flickr, you go here:

```
http://www.flickr.com/invite.gne
```

Groups

Groups allow people to organize themselves into communities based around themes, places, and common interests. Take a look at all the groups that are in Flickr:

```
http://www.flickr.com/groups/
```

You access an individual group here:

```
http://www.flickr.com/groups/{group-id}/
```

where `group-id` is the NSID of the group or its friendly name, which the group owner sets here:

```
http://www.flickr.com/groups_url.gne?id={group-nsid}
```

Consider, for instance, the Flickr Central Group, which is accessed from here:

```
http://www.flickr.com/groups/34427469792@N01/
```

and from here:

```
http://www.flickr.com/groups/central/
```

You can page through the discussion in a group here:

```
http://www.flickr.com/groups/{group-id}/discuss/page{page-number}/
```

You can post a new topic here:

```
http://www.flickr.com/groups_newtopic.gne?id={group-nsid}
```

For example:

```
http://www.flickr.com/groups_newtopic.gne?id=34427469792@N01
```

You access a specific thread here:

```
http://www.flickr.com/groups/{group-id}/discuss/{thread-id}/
```

For example:

```
http://www.flickr.com/groups/central/discuss/140537/
```

You access a specific comment in the thread here:

```
http://www.flickr.com/groups/{group-id}/discuss/{thread-id}/#comment{comment-id}
```

For example:

```
http://www.flickr.com/groups/central/discuss/140537/#comment1192964
```

You can edit, delete, or lock a thread if you have the appropriate rights:

```
http://www.flickr.com/groups/{group-id}/discuss/{thread-id}/{thread-action}
```

where `{thread-action}` is `edit`, `delete`, or `lock`.

Similarly, for the comments that hang off a thread (one-deep), you can find them here:

```
http://www.flickr.com/groups/{group-id}/discuss/{thread-id}/{comment-id}/
{comment-action}/
```

where `{comment-action}` can be `edit` or `delete`.

Each group has a photo pool accessible here:

```
http://www.flickr.com/groups/{group-id}/pool/
```

For example:

```
http://www.flickr.com/groups/central/pool/
```

You can look at the geotagged photos from the group on a map here:

```
http://www.flickr.com/groups/{group-id}/pool/map?mode=group
```

You can look at a list of the most popular tags used for photos in a group here:

```
http://www.flickr.com/groups/{group-id}/pool/tags/
```

You can look at photos with a certain `tag` in the group here:

```
http://www.flickr.com/groups/{group-id}/pool/tags/{tag}/
```

You can look at photos that have been contributed to the pool by a specific user here:

```
http://www.flickr.com/groups/{group-id}/pool/{user-nsid}/
```

Account Management

Some URLs are used for account management functions. You need to be logged in to access them.

To access your contacts' photos, go here:

```
http://www.flickr.com/photos/friends/
```

To manage your account, go here:

```
http://www.flickr.com/account
```

You can adjust various specific options at the following URLs:

```
http://www.flickr.com/account?tab=e-mail
http://www.flickr.com/account/?tab=privacy
http://www.flickr.com/account/?tab=extend
http://www.flickr.com/account/order/history/
http://www.flickr.com/account/prefs/screenname/
http://www.flickr.com/account/prefs/layout/
```

Browsing Through Flickr

Flickr's jumping-off point for looking at the world of Flickr is this:

```
http://www.flickr.com/explore/
```

To look at what Flickr rates as the most "interesting" photos, go here:

```
http://www.flickr.com/explore/interesting/
```

The page gives some sense of how Flickr rates interestingness (even if there aren't complete details given):

> *There are lots of things that make a photo "interesting" (or not) in the Flickr: where the clickthroughs are coming from, who comments on it and when, who marks it as a favorite, its tags, and many more things which are constantly changing. Interestingness changes over time, as more and more fantastic photos and stories are added to Flickr.*

You can look at the photos the most interesting photos for a specific period of time. A special case is a random selection of photos from the last seven days:

```
http://www.flickr.com/explore/interesting/7days/
```

You can see interesting photos for a given month or day, the latter as a calendar or slide show:

```
http://www.flickr.com/explore/interesting/{year}/{month}/
http://www.flickr.com/explore/interesting/{year}/{month}/{day}
http://www.flickr.com/explore/interesting/{year}/{month}/{day}/show/
```

For example:

```
http://www.flickr.com/explore/interesting/2007/01/04/show/
```

Search

Flickr provides interfaces for basic and advanced photo searches.

Basic Photo Search

The photo search URL is constructed as follows:

```
http://www.flickr.com/search/?w={search-scope}&q={search-term}&m={search-mode}
```

where `search-scope` is one of `all`, `faves`, or the `{user-id}` of a user and where `search-mode` is `tags` or `text`. You can use some optional parameters to qualify the search:

* `&z=t` for thumbnails (as opposed to the detail view)
* `&s=int` or `&s=rec` to sort by interestingness or by recent date
* `&page={page-number}` to page through the results

Advanced Photo Search

For the advanced photo search (`http://www.flickr.com/search/advanced`), you can figure out other ways to modify the search URL.

You can add terms to `{search-term}` by adding a hyphen (`-`) before the term. For instance, you can look for photos that are tagged with `flower` but not `rose` or `tulip` with this:

```
http://www.flickr.com/search/?q=flower+-rose+-tulip&m=tags&ct=0
```

You can use add `safe-search` options with this:

```
&ss={safe-search}
```

where `{safe-search}` is `0`,`1`, or `2` corresponding to `on`, `moderate`, and `off`, respectively.

You can limit searches to a particular `content-type` by using this:

```
&ct={content-type}
```

where `{content-type}` is one of the following:

* 0 for photos
* 1 for screenshots
* 2 for other stuff (art, drawings, CGI, and so on)
* 3 for photos and screenshots
* 4 for screenshots and other stuff
* 5 for photos and other stuff
* 6 for photos and other stuff and screenshots

You can also limit photos by a date range:

```
&d={taken-or-posted}-{from-date}-{to-date}
```

where `{taken-or-posted}` is `taken` or `posted` and where `{from-date}` and `{to-date}` are of the form `yyyymmdd`. You can state one or both of the dates. For example:

```
&d=posted--20070702
&d=taken-20070613-20070702
```

Finally, you can search for photos with certain Creative Commons licenses by using this:

```
&l={CC-license}
```

where `{CC-license}` can be one of `cc` (for any Creative Commons license), `com` (for licenses that permit commercial reuse), or `deriv` (for licenses that permit derivative works).

Note I do not provide a full analysis of the URL language for searching groups (`http://flickr.com/search/groups/`) and users (`http://flickr.com/search/people/`).

Geotagged Photos in Flickr

You can use the Flickr World map to plot georeferenced photos here:

```
http://www.flickr.com/map/
```

You can control the center, zoom level, and display type of the map with this:

```
http://www.flickr.com/map/?&fLat={lat}&fLon={lon}&zl={zoom-level}&
map_type={map-type}
```

where `zoom-level` is an integer ranging from `1` to `17` (`17` is the most zoomed out) and `map-type` is `hyb` or `sat`. If `map-type` is not explicitly set, the map has a default (political-style) map.

You can filter photos in various ways by adding more parameters to the URL:

* By search terms with this:

```
&q={search-term}
```

* By group with this:

```
&group_id={group-nsid}
```

* By person with this:

```
&user_id={user-nsid}
```

* By date bounds where `taken-date` is of the form `yyyy-mm-dd%20hh:mm:ss`:

```
&min_taken_date={taken-date}
&max_taken_date={taken-date}
```

and with the following:

```
&min_upload_date={upload-date}
&max_upload_date={upload-date}
```

where `upload-date` is a Unix timestamp (number of seconds since January 1, 1970, UTC).

* By page with this:

```
&page={page-number}
```

* By interestingness with this:

```
&s=int
```

For example, this address:

```
http://www.flickr.com/map/?&q=flower&fLat=37.871268&fLon=-122.286414&zl=4
```

produces a map of geotagged pictures around Berkeley, California, filtered on a full-text search of `flower`. A corresponding list view according to Flickr is as follows:

```
http://www.flickr.com/search/?&q=flower&m=text&s=rec&b=-122.346496,37.847598,
-122.226333,37.894938&a=10&d=taken-19700101-
```

This search uses parameters I have already presented in the “Advanced Photo Search” section in addition to this for a geographic bounding box:

```
&b={lon0},{lat0}{lon1},{lat1}
```

and this:

```
&a={accuracy}
```

where `accuracy` is presumably the same parameter as the accuracy parameter used in the Flickr API in `flickr.photo.search` to denote the "recorded accuracy level of location information."[6]

6. http://www.flickr.com/services/api/flickr.photos.search.html

The Flickr Organizer

You can use the JavaScript-based Organizer to process your Flickr photos:

```
http://www.flickr.com/photos/organize/
```

Most of its functionality is not addressable through URLs, but a few aspects are. You can process your recently uploaded photos here:

```
http://www.flickr.com/photos/organize/?start_batch=recent_uploads
```

You can create and organize your sets and collections here:

```
http://www.flickr.com/photos/organize/?start_tab=sets
```

Finally, you can tag your untagged photos here:

```
http://www.flickr.com/photos/organize/?start_batch=untagged&mode=together
```

Recent Activities

You can look at recent activities around your photos here:

```
http://www.flickr.com/recent_activity.gne?days={time-period}
```

where `time-period` can be any of the following:

* A natural number (up to some limit that I've not tried to determine) to indicate the number of days
* A natural number appended with `h` for number of hours
* Blank to mean "since last login"

Mailing Interfaces

Flickr has its own e-mail type interface for facilitating communication among Flickr people:

```
http://www.flickr.com/messages.gne?ok=1
```

This messaging facility allows communication by proxy to retain the anonymity of users. You can access your sent mail here:

```
http://www.flickr.com/messages_sent.gne
```

You can read a message here:

```
http://www.flickr.com/messages_read.gne?id={message-id}
```

You compose a new message here:

```
http://www.flickr.com/messages_write.gne
```

Interfacing to Weblogs

One fun thing to do with pictures is to send a picture to one's weblog along with some commentary. Flickr helps make the process easier to do. You configure weblogs here:

```
http://www.flickr.com/blogs.gne
```

You can configure the settings for a specific weblog here:

```
http://www.flickr.com/blogs_edit.gne?id={blog-id}
```

You can configure the layout here:

```
http://www.flickr.com/blogs_layout.gne?id={blog-id}&edit=1
```

In Chapter 5, I go into greater detail about how the properties used to set up a blog to work with Flickr is a reflection of the blogging APIs that you will study.

Syndication Feeds: RSS and Atom

RSS and Atom feeds are well integrated in Flickr. These feeds are an example of XML, and you will learn more about that in Chapter 4. Flickr implements RSS and other syndication feeds in an extensive manner, as documented here:

```
http://www.flickr.com/services/feeds/
```

There's a lot to cover, which I'll come back to in Chapter 4.

Mobile Access

Flickr provides a model to help you integrate your own services with mobile devices. For example, you can e-mail pictures to Flickr. This functionality is not strictly tied to mobile devices but is particularly useful on a mobile phone because e-mail is perhaps the most convenient way to upload a picture from a camera phone while away from your desk. You can configure e-mail uploading here:

```
http://www.flickr.com/account/uploadbye-mail/
```

You can also look at pictures on a mobile device through a simplified interface customized for small displays here:

```
http://m.flickr.com
```

Third-Party Flickr Apps

Flickr has an API that enables the development of third-party applications or tools. The API is at the heart of what makes Flickr such a great mashup platform. Hundreds of third-party apps have been written to use the API, and these apps have made it easier and more fun and surprising to use Flickr. The Google Maps and Flickr Greasemonkey script are examples of third-party Flickr apps.

Go to http://www.flickr.com/services/ to see a list of such third-party Flickr apps. To get inspired about how you can use the Flickr API in fun, useful, and imaginative ways, play with John Watson's collections of Flickr Toys:

```
http://bighugelabs.com/flickr/
```

While looking at the Flickr Toys, think about what content and services from Flickr are being accessed.

I analyze various Flickr third-party applications in more detail in conjunction with my study of the Flickr API in Chapter 6.

Creative Commons Licensing

Under copyright laws in the United States, you can't reuse other people's pictures by default except under the "fair use" rule. If someone uses a Creative Commons (CC) license for a picture, the owner is saying, "Hey, you can use my picture under looser restrictions without having to ask me for permission." You can see a license attached to any given picture.

Flickr makes it easy for users to associate CC licenses with their photos. You can browse and search for photos by CC license here:

```
http://www.flickr.com/creativecommons/
```

You can look at pictures by specific license here:

```
http://www.flickr.com/creativecommons/{cc-license}/
```

where `{cc-license}` is currently one of the following:

* `by-2.0`
* `by-nd-2.0`
* `by-nc-nd-2.0`
* `by-nc-2.0`
* `by-nc-sa-2.0`
* `by-sa-2.0`

Consult the following to get an understanding of the various licenses:

```
http://creativecommons.org/licenses/
```

Cameras

Flickr enables users to view photos by the brand of cameras used to take them:

```
http://www.flickr.com/cameras/
```

To get at all the brands of cameras, see the following:

```
http://www.flickr.com/cameras/brands/
```

You can look at pictures by the specific camera type here:

```
http://www.flickr.com/cameras/{camera-company}/{camera-model}/
```

For example:

```
http://www.flickr.com/cameras/canon/
http://www.flickr.com/cameras/canon/powershot_sd600/
```

The Mashup-by-URL-Templating-and-Embedding Pattern

Let's now apply Flickr's URL language to make a simple mashup with Flickr. In this section, I'll show how to create a simple example of what I call the Mashup-by-URL-Templating-and-Embedding pattern. Specifically, I connect Flickr archives and a WordPress weblog by virtue of translating URLs; an HTML page takes a given year and month and displays my Flickr photos along with the entries from the weblog for this book (http://blog.mashupguide.net). The mashup works because both the Flickr archives and the entries for the weblog are addressable by year and month. For Flickr, recall the following URL template for the archives:

```
http://www.flickr.com/photos/{user-id}/archives/{date-taken-or-posted}/
{year}/{month}/{archive-view}
```

For example:

```
http://www.flickr.com/photos/raymondyee/archives/date-taken/2007/06/calendar/
```

The weblog has URLs for posts by year and month (if posts from those dates exist):

```
http://blog.mashupguide.net/{year}/{month}
```

For example:

```
http://blog.mashupguide.net/2007/06/
```

The mashup takes the year and month from the user and loads two iframes corresponding to the Flickr photos and Mashupguide.net entries for the month by constructing the URLs for the year and month:[7]

7. http://examples.mashupguide.net/ch02/Flickr.and.WordPress.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <title>Raymond Yee's Flickr and mashupguide weblog</title>
    <script type="text/javascript">
      //<![CDATA[
      function reloadFrames() {
        // get a handle to the iframes and the year and month in the form
        var dateForm = document.getElementById('date');
        var flickrFrame = document.getElementById('FlickrFrame');
        var wpFrame = document.getElementById('WPFrame');
        year = dateForm.year.value;
        month = dateForm.month.value;
        var year, month, dateForm;
        var flickrURL =
          "http://www.flickr.com/photos/raymondyee/archives/date-taken/" +
```

```
            year + "/" + month + "/calendar";
        var wpURL = "http://blog.mashupguide.net/" + year + "/" + month + "/";
        //reset the URLs for the iframes
        flickrFrame.src = flickrURL;
        wpFrame.src = wpURL;
        return false;
      }
      //]]>
    </script>
  </head>
  <body>
    <form id="date" action="#" onsubmit="return reloadFrames();">
      Year: <input type="text" size="4" name="year" value="2007" />
      Month: <input type="text" size="4" name="month" value="06" />
      <input type="submit" value="Reload Frames" />
    </form>
    <iframe id="FlickrFrame"
            src="http://www.flickr.com/photos/raymondyee/archives/date-
taken/2007/
06/calendar/"
            name="Flickr" style="width:600px; height:500px; border:
0px"></iframe>
    <iframe id="WPFrame" src="http://blog.mashupguide.net/2007/06/"
            name="WordPress"
    style="width:600px; height:500px; border: 0px"></iframe>
  </body>
</html>
```

This example may seem trivial, even accounting for its intentional simplicity as an illustration, but ask yourself, what if you wanted to add a third source, such as the posts I made to del.icio.us posts for a given month? As you will see later in this chapter, there is no delic.io.us URL corresponding to a listing of the bookmarks uploaded in a given year or month (that is, the human UI to del.icio.us is not addressable by the posting date), so I can't add del.icio.us to my mashup by adding a corresponding iframe and URI template. Addressability of resources is what makes the Mashup-by-URL-Templating-and-Embedding pattern possible.

Note You can use `https://api.del.icio.us/v1/posts/dates` to get a list of the number of posts for a date and then use `https://api.del.icio.us/v1/posts/get?` to retrieve them. You can configure del.icio.us to send your daily postings to your blog (`https://secure.del.icio.us/settings/user-id/blogging/posting`).

Granular URI addressability, the ability to refer to resources through a URI in very specific terms, enables simple mashups. This is especially true if the parameters in the URI templates are ones that have the same meaning across many web sites. Such identifiers are often the point of commonality between URIs from different sites. You have seen a number of such identifiers already:

* ISBN

* Year, month, day
* Latitude and longitude
* URLs themselves; for example, `http://validator.w3.org?uri={uri-to-validate}`, where uri-to-validate is a URL to validate, such as `http://validator.w3.org/check?uri=http%3A%2F%2Fvalidator.w3.org%2F`)

These identifiers contrast with application-specific identifiers (such as NSIDs of Flickr users and groups). Somewhere between widely used identifiers and those that are confined to one application only are objects such as tags, which may or may not have meaning beyond the originating web site. I'll return to this issue in Chapter 3.

Google Maps

Now, let's turn to studying the functionality of Google Maps, located at `http://maps.google.com/`.

With the standard Google Maps site, you can do the following:

* You can search for locations on a map.
* You can search for businesses on a map.
* You can get driving directions between two points.
* You can make your own map now with the My Maps feature.

You can also embed a Google Maps "widget" into a web page via JavaScript—using the Google Maps API.[8] The focus of this chapter is on maps that are hosted directly by Google. I examine third-party embedded Google maps in Chapters 8 and 13.

Even though Google Maps is not the most highly trafficked online map site,[9] it is (according to Programmableweb.com), the application is often used in mashups.

8. http://www.google.com/apis/maps/

9. http://news.yahoo.com/s/ap/20070405/ap_on_hi_te/google_maps—"Google's maps already are a big draw, with 22.2 million U.S. visitors during February, according to the most recent data available from comScore Media Metrix. That ranked Google Maps third in its category, trailing AOL's Mapquest (45.1 million visitors) and Yahoo (29.1 million visitors)."

URL Language of Google Maps

Understanding the syntax and semantics of URLs in Google Maps will help you better recombine the functionality of the standard Google Maps site. Consider an example: I have an address I want to locate—for instance, the address of the White House (1600 Pennsylvania Ave., Washington, D.C.). I go to Google Maps (`http://maps.google.com/`) and type **1600 Pennsylvania Ave, Washington, DC** into the search box to get a map. I get the URL for the map by examining the "Link to this page" link:

```
http://maps.google.com/maps?f=q&hl=en&q=1600+Pennsylvania+Ave,+Washington,+DC&
sll=36.60585,-121.858956&sspn=0.006313,0.01133&ie=UTF8&z=16&om=1&iwloc=addr
```

What do the various parameters in the URL mean? Table 2-4 draws from the Google Maps Parameters page of the Mapki wiki.[10]

Table 2-4. Dissecting Parameters for a Link to Google Maps

Parameter	Description
f=q	The f parameter, which controls the display of the Google Maps form, can be d (for the directions form or l for the local form). Without the f parameter, the default search form is displayed.
hl=en	Google Maps supports a limited number of host languages, including en for English and fr for French.
q=1600+Pennsylvania+Ave, entered +Washington,+DC	The value of the q parameter is treated as though it were via the query box at http://maps.google.com.
sll=36.60585,	sll contains the latitude and longitude for the center point around -121.858956 which a business search is performed.
spn=0.006313,	spn is the approximate latitude/longitude span for the map.0.01133
ie=UTF8	ie is the character encoding for the map.
om=1	om determines whether to include an overview map. With om=0, the overview map is closed.
iwloc=addr	iwloc controls display options for the info window.

A good way to get a feel for how these parameters function is to change a parameter, add new ones, or drop ones in the sample URL and take a look at the resulting map. For instance, if you have only the q parameter, you would still get a map with some default behavior:

```
http://maps.google.com/maps?q=1600+Pennsylvania+Ave,+Washington,+DC
```

That is, the other parameters are not mandatory. Let's play with the z parameter to adjust the zoom factor:

```
http://maps.google.com/maps?q=1600+Pennsylvania+Ave,+Washington,+DC&z=0
```

versus the following:

```
http://maps.google.com/maps?q=1600+Pennsylvania+Ave,+Washington,+DC&z=17
```

There is a comprehensive list of Google Maps parameters[11] to help you figure out the common and uncommon parameters. Since the wiki page is not part of Google's documentation, you can't take it as an official description of the URL language of Google Maps. However, the web page is also the work of a highly engaged community, actively working on uncovering every nook and cranny of Google Maps. With the list of parameters, you can learn some features that you might not have known from casual use of the Google Maps user interface. For instance:

10. http://mapki.com/wiki/Google_Map_Parameters, accessed as http://mapki.com/index.php?title=Google_Map_Parameters&oldid=4145

11. http://mapki.com/wiki/Google_Map_Parameters, accessed as http://maps.google.com/maps?f=q&hl=en&q=1600+Pennsylvania+Ave,+Washington,+DC on April 14, 2007

* mrad lets you specify an additional destination address.
* output=kml gets a KML file to send to Google Earth.
* layer=t adds the traffic layer.

* `mrt=kmlkmz` shows "user-created content." For example, the following shows user-generated information about hotels around the White House:

```
http://maps.google.com/maps?f=q&hl=en&q=hotel&near=1600+Pennsylvania+Ave,
+Washington,+DC&sll=36.60585,-
121.858956&sspn=0.006313,0.01133&ie=UTF8&z=16&om=1&
iwloc=addr&mrt=kmlkmz
```

Just as you can create mashups involving Flickr by using Flickr's URL language, you can create mashups with Google Maps by exploiting its URL structures. Let's consider a few examples.

Viewing KML Files in Google Maps

Many of the popular sources for KML (such as `http://earth.google.com/gallery/`) assume you will view KML in Google Earth. However, you can display a limited subset of KML in Google Maps. Consider, for instance, the KML file at the following location:

```
http://services.google.com/earth/kmz/global_heritage_fund_n.kmz
```

It can be viewed in Google Maps by passing in the URL of the KML file via the `q` parameter, as shown here:

```
http://maps.google.com/maps?q=http:%2F%2Fservices.google.com%2Fearth%2Fkmz%2Fglo
bal_
heritage_fund_n.kmz
```

Hence, in your own web site, you can give the option to your users of downloading KML to Google Earth or viewing the KML on Google Maps by linking to the following:

```
http://maps.google.com/maps?q={URL-of-KML}
```

Connecting Yahoo! Pipes and Google Maps

A specific case of displaying KML files is feeding KML from Yahoo! Pipes into Google Maps. (I describe Yahoo! Pipes in detail in Chapter 4. For the purposes of this discussion, you need to know only that Yahoo! Pipes can generate KML output.) Consider, for example, Apartment Near Something, configured specifically to list apartments that are close to cafes around UC Berkeley:

```
http://pipes.yahoo.com/pipes/pipe.info?location=94720&what=cafes&mindist=2&
=Run+Pipe&_id=1mrlkB232xGjJDdwXqIxGw&_run=1
```

You can get KML output from Yahoo! Pipes from the following:

```
http://pipes.yahoo.com/pipes/pipe.run?_id=1mrlkB232xGjJDdwXqIxGw&_render=kml&_ru
n=1&
location=94720&mindist=2&what=cafes
```

which you can feed into Google Maps in the `q={URL-of-KML}` parameter:

```
http://maps.google.com/maps?f=q&hl=en&geocode=&q=http%3A%2F%2Fpipes.yahoo.com%2F
pipes%2Fpipe.run%3F_id%3D1mrlkB232xGjJDdwXqIxGw%26_render%3Dkml%26_run%3D1%26
location%3D94720%26mindist%3D2%26what%3Dcafes&ie=UTF8&ll=37.992916,-122.24556&
```

```
spn=0.189398,0.362549&z=12&om=1
```

Other Simple Applications of the Google Maps URL Language

Here are a few other examples of how to connect Google Maps to your applications by creating the appropriate URL:

* Let's not forget that by just using `q={address}`, you can now generate a URL to a map centered around that address. If such a map suffices, it's hard to imagine a simpler way to create a map corresponding to that address. No geocoding is needed.
* You can create a URL for custom driving directions for any source and destination address creating custom driving directions from your spreadsheet of addresses by making the URLs. For example, to generate driving directions from Apress to the Computer History Museum, you can use this:

```
http://www.google.com/maps?saddr={source-address}&daddr=
{destination-address}
```

to generate this:

```
http://www.google.com/maps?saddr=2855+Telegraph+Ave,+Berkeley,+CA+
94705&daddr=1401+N+Shoreline+Blvd,+Mountain+View,+CA+94043
```

Although driving directions have recently been added to the Google Maps API,[12] it is currently not possible to use the API to create directions to avoid highways, something you can do by using the `dirflg=h` parameter.[13] Hence, you can easily generate a scenic route for myself between the Apress offices and the Computer Museum, while avoiding the API altogether:

12. http://www.google.com/apis/maps/documentation/#Driving_Directions

13. http://groups.google.com/group/Google-Maps-API/browse_thread/thread/279ee413e4e0309/0dabfb71863af712?lnk=gst&q=avoid+highway&rnum=2#0dabfb71863af712

```
http://www.google.com/maps?saddr=2855+Telegraph+Ave,+Berkeley,+CA
+94705&daddr=1401+N+Shoreline+Blvd,+Mountain+View,+CA+94043&dirflg=h
```

It pays to know the URL language of an application!

* You can use Google Maps as a nonprogrammer's geocoder. Center the map on the point for which you want to calculate its latitude and longitude, and read the values off the `ll` parameter. If the `ll` parameter is not present, you can double-click the center of the map, just enough to cause the map to recenter on the requested point.

Amazon

Amazon is the third major example in this chapter. Not only is Amazon a popular e-commerce site, but it is an e-commerce platform this is easily remixed with other content. Although you will study the Amazon APIs later in this book, you'll focus here

on Amazon from the view of an end user. Moreover, the goal in this section is not to learn all the features of Amazon but rather to study its URL language.

Note Although Amazon sells merchandise other than books, I use books in my examples. Moreover, I focus on Amazon, the site geared to the United States instead of Amazon's network of sites aimed to customers outside the United States.

The strategy you'll follow here is to discern the key entities of the Amazon site through a combination of using and experimenting with the site, sifting through documentation, and seeing what other users have done. You will see that figuring out the structure of Amazon's URLs is not as straightforward as working through the Flickr URL language. Since some of the conclusions here are not supported by official documentation from Amazon, I cannot make any long-term guarantee behind the URLs.

Amazon Items

It doesn't take much analysis of Amazon to see that the central entity of the site is an item for sale (akin to a photo in Flickr). By looking at the URL of a given item and looking throughout a page describing it, you will see that Amazon uses an Amazon Standard Identification Number (ASIN) as a unique identifier for its products.[14] For books that have an ISBN, the ASIN is the same as the ISBN-10 for the book. According to the Wikipedia article on ASIN, you can point to a product with an ASIN with the following URL:

```
http://www.amazon.com/gp/product/{ASIN}
```

Take for instance, Czesl´aw Mil´osz's *New and Collected Poems* (paperback edition), which has an ISBN-10 of 0060514485. You can find it on Amazon here:

```
http://www.amazon.com/gp/product/0060514485
```

It is important to know that the way to link to Amazon has changed in the past and will likely continue to change. For instance, you can also link to the book with this:

```
http://www.amazon.com/exec/obidos/ASIN/0060514485
```

or even with this shorter form:

```
http://amazon.com/o/ASIN/0060514485
```

14. http://en.wikipedia.org/wiki/Amazon_Standard_Identification_Number

Using this syntax would ideally be founded on some official documentation from Amazon. Where would you find definitive documentation on how to structure a link to a product of a given ASIN? My search through the Amazon developers' site led to the technical documentation,[15] whose latest version at the time of writing was the April 4, 2004, edition.[16] That trail leads ultimately to a page on the use of identifiers, which, alas, does not spell out how to formulate the URL for an item with a given ASIN.[17] The bottom line for now is that Wikipedia, combined with experimentation, is the best way to discern the URL structures of Amazon.

Let's apply this approach to other functions of Amazon. For instance, can you generate a URL for a full-text search? Go to Amazon, and enter your favorite search term. Take for example, `flower`. When I hit Submit, I got the following URL:

```
http://amazon.com/s/ref=nb_ss_gw/102-1755462-2944952?url=search-
alias%3Daps&field-
keywords=flower&Go.x=0&Go.y=0
```

If I did the search again, say in a different browser, I got another URL:

```
http://amazon.com/s/ref=nb_ss_gw/102-8204915-1347316?url=search-
alias%3Daps&field-
keywords=flower&Go.x=0&Go.y=0&Go=Go
```

Notice where things are similar and where they are different. Looking for what's common (the `http://amazon.com/s` prefix and the `?url=search-alias%3Daps&field-keywords=flower&Go.x=0&Go.y=0&Go=Go` argument), I eliminated the sections that were different to get the following:

```
http://amazon.com/s/?url=search-alias%3Daps&field-keywords=flower&Go.x=0&Go.y=0&
Go=Go
```

This URL seemed to work fine. You can even eliminate `&Go.x=0&Go.y=0&Go=Go` to boil the request down to this:

```
http://amazon.com/s/?url=search-alias%3Daps&field-keywords=flower
```

So, how do you limit the search to books? Going to Amazon, selecting the Book section, and using the `flower` keyword, you can get to the following URL:

```
http://amazon.com/s/ref=nb_ss_gw/102-6984159-2338509?url=search-
alias%3Dstripbooks&field-keywords=flower&Go.x=12&Go.y=6
```

Stripping away the parameters as before gave me this:

```
http://amazon.com/s/?url=search-alias%3Dstripbooks&field-keywords=flower
```

This trick works for the other departments. For example, to do a search on `flowers` in Home & Garden, use this:

```
http://amazon.com/s/?url=search-alias%3Dgarden&field-keywords=flower
```

Based on these experiments, I would conclude that the URL for searching for a keyword in a department is as follows:

```
http://amazon.com/s/?url=search-alias%3D{amazon-dept}&field-keywords={keyword}
```

Let's run through the syntax of other organizational structures.

15. http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=19

16. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=703&categoryID=19

17. http://docs.amazonwebservices.com/AWSECommerceService/2007-04-04/DG/ItemIdentifiers.html

Lists

You can find the Wish List section at the following URL:

```
http://www.amazon.com/gp/registry/wishlist/
```

If you are logged in, you will see a list of your lists on the left. Look at the URL of one of the lists, which will look something like the one for my public wish list:

```
http://www.amazon.com/gp/registry/wishlist/1U5EXVPVS3WP5/ref=cm_wl_rlist_go/102-5889202-4328156
```

Now look at another. I surmised that since the number on the right (`102-5889202-4328156`) remained the same and the other number (`1U5EXVPVS3WP5`) changed for each list, the unique `1U5EXVPVS3WP5` is the identifier for the list. You can point to a list using its list identifier by entering something similar to the following:

```
http://www.amazon.com/gp/registry/wishlist/1U5EXVPVS3WP5
```

Hence, you can conclude that the URL for a wish list is as follows:

```
http://www.amazon.com/gp/registry/wishlist/{wishlist-id}
```

Tags

Tags are a recent introduction to Amazon. You will see links like the following:

```
http://www.amazon.com/tag/czeslaw%20milosz/ref=tag_dp_ct/102-8204915-1347316
```

which can be reduced (following the strategy you took for other parts of Amazon) to this:

```
http://www.amazon.com/tag/czeslaw%20milosz/
```

The URL for books that correspond to a tag is as follows:

```
http://www.amazon.com/tag/{tag}/
```

Subject Headings

In looking through the Browse Subject section of Amazon (`http://www.amazon.com/Subjects-Books/b/?ie=UTF8&node=1000`), you can find a link such as the following:

```
http://www.amazon.com/b/ref=amb_link_1760642_21/104-0367717-9318361?ie=UTF8&node=5&pf_rd_m=ATVPDKIKX0DER&pf_rd_s=center-3&pf_rd_r=0J0MADE0YSN1VRBA6XZS&pf_rd_t=101&pf_rd_p=233185601&pf_rd_i=1000
```

This refers to the Computers & Internet Section, which you can reduce to the following:

```
http://www.amazon.com/b/?ie=UTF8&node=5
```

from which you can conclude that the URL for a section is as follows:

```
http://www.amazon.com/b/?ie=UTF8&node={node-number}
```

Caution The fact that the node is specified by number corresponding to its order by alphabetical listing rather than a unique key makes me concerned about the long-term stability of the link. Will 5 always refer to computers, or if there is another section added that goes before it alphabetically, will the link break?

There are plenty of other entities whose URL structures can be discerned, including the following:

* Listmania lists:

```
http://www.amazon.com/lm/{list-mania-id}/
```

For example:

```
http://www.amazon.com/lm/1FHOE3G892IA/
```

* So You'd Like To guides:

```
http://www.amazon.com/gp/richpub/syltguides/fullview/{slltg-id}
```

For example:

```
http://www.amazon.com/gp/richpub/syltguides/fullview/3T3I3YDBG889B
```

* Personal profiles:

```
http://www.amazon.com/gp/pdp/profile/{user-id}/
```

For example:

```
http://www.amazon.com/gp/pdp/profile/A2D978B87TKMS2/
```

* Similar items for a book:

```
http://amazon.com/sim/{ISBN-10}/1/ref=pd_sexpl_esi/
```

For example:

```
http://amazon.com/sim/0060514485/1/ref=pd_sexpl_esi/
```

In Chapter 1, you already studied how the Amazon URL language is used by the LibraryLookup bookmarklet to mash up Amazon and library catalogs. Linking to Amazon resources by a tag enables tag-based mashups, which I will describe in Chapter 3.

del.icio.us

The social-bookmarking site del.icio.us is a deeply influential site, credited by many for jump-starting the immense amount of activity around tagging.

The main resources of importance in del.icio.us (`http://del.ico.us`) are bookmarks, that is, URLs. You can associate tags with a given URL and look at an individual's collection of URLs and the tags they use. In this section, I again explain the URL structures by browsing through the site and noting the corresponding URLs.

You can look at the public bookmarks for a specific user (such as rdhyee) here:

```
http://del.icio.us/{user-id}
```

For example:

```
http://del.icio.us/rdhyee
```

You can see all the bookmarks of a user by tag here:

```
http://del.icio.us/{user-id}/{tag}
```

For example:

```
http://del.icio.us/rdhyee/NYTimes
```

You can see all the URLs that people have tagged with a given tag here:

```
http://del.icio.us/tag/{tag}
```

You can see just the popular URLs associated with the tag here:

```
http://del.icio.us/popular/{tag}
```

You can access today's popular items here:

```
http://del.icio.us/popular/
```

You can access just the newest popular ones here:

```
http://del.icio.us/popular/?new
```

Correlating a URL to a del.icio.us page is a bit trickier. Consider the following URL:

```
http://harpers.org/TheEcstasyOfInfluence.html
```

which you can reference from del.icio.us here:

```
http://del.icio.us/url/53113b15b14c90292a02c24b55c316e5
```

So, how do you get `53113b15b14c90292a02c24b55c316e5` from `http://harpers.org/TheEcstasyOfInfluence.html`? The answer is that the identifier is an md5 hash of the URL. In Python, the following line of code:

```
md5.new("http://harpers.org/TheEcstasyOfInfluence.html").hexdigest()
```

or the following PHP code:

```
<?php
 $url = "http://harpers.org/TheEcstasyOfInfluence.html";
 print md5($url);
?>
```

yields `53113b15b14c90292a02c24b55c316e5`.

Note that the following:

```
http://del.icio.us/url?url=http://harpers.org/TheEcstasyOfInfluence.html
```

also does work and redirects to the following:

```
http://del.icio.us/url/53113b15b14c90292a02c24b55c316e5
```

Screen-Scraping and Bots

The focus of this book is on creating mashups using public APIs and web services. If you want to mash up a web site, one of the first things to look for is a public API. A public API is specifically designed as an official channel for giving you programmatic access to data and services of the web site. In some cases, however, you may want to create mashups of services and data for which there is no public API. Even if there is a public API, it is extremely useful to look beyond just the API. An API is often incomplete. That is, there is functionality in the user interface that is not included in the API. Without a public API for a web site, you need to resort to other techniques to reuse the data and functionality of the application.

One such technique is screen-scraping, which involves extracting data from the user interface designed for display to human users. Let me define bots and spiders, which often use screen-scraping techniques. Bots (also known as an *Internet bots*, *web robots*, and *webbots*) are computer programs that "run automated tasks over the Internet," typically tasks that are "both simple and structurally repetitive."[18] Bots come in a variety of well-known types and engage in activities that range from positive and benign to illegal and destructive:

* "Chatterbots" that automatically reply to human users through instant messaging or IRC[19]
* Wikipedia bots that automate the monitoring, maintaining, and editing of the Wikipedia[20]
* Ticket-purchasing bots that buy tickets on behalf of ticket scalpers
* Bots that generate spam or launch distributed denial of service attacks

Web spiders (also known as *web crawlers* and *web harvesters*) are a special type of Internet bot. They typically focus on getting collections of web pages—up to billions of pages—rather than focused extraction of data on a given page. It's the spiders from search engines such as Google and Yahoo! that visit your web pages to collect your web pages with which to build their large indexes of the Web.

There are some important technical challenges to screen-scraping. The vast majority of data embedded in HTML is not marked up to be unambiguously and consistently parsed by bots. Hence, screen-scraping depends on making rather brittle assumptions about what the placement and presentation style of embedded data implies about the semantics of the data. The author of web pages often changes its visual style without intending to change any underlying semantics—but still ends up breaking, often inadvertently, screen-scraping code. In contrast, by packaging data in commonly understood formats such as XML geared to computer consumption, you are an implicit—if not explicit—commitment to the reliable transfer of data to others. Public API functions are controlled, defined programmatic interfaces between the creator of the site and you as the user. Hence, accessing data through the public API should theoretically be less fragile than screen-scraping/web-scraping a web site.

18. http://en.wikipedia.org/wiki/Internet_bot, accessed on July 11, 2007, as http://en.wikipedia.org/w/index.php?title=Internet_bot&oldid=142845374

19. http://en.wikipedia.org/wiki/Chatterbot

20. http://en.wikipedia.org/wiki/Wikipedia:Bots

Caution Since I'm not a lawyer, do not construe anything in this book, including the following discussion, as legal advice!

If you engage in screen-scraping, you need to be thoughtful about how you go about it and, in some cases, even whether you should do it in the first place. Start with reading the terms of service (ToS) of the web site. Some ToSs explicitly forbid the use of bots (such as automated crawling) of their sites. How should you respond to such terms of services? On the one hand, you could decide to take a conservative stance and not

screen-scrape the site at all. Or you could go to the other extreme and screen-scrape the site at will, waging that you won't get sued and noting that if the web site owner is not happy, the owner could just use technical means to shut down your bot.

I think a middle ground is often in order, one that is well-stated by Bausch, Calishan, and Dornfest: "So use the API whenever you can, scrape only when you absolutely must, and mind your Ps and Qs when fiddling about with other people's data."[21] In other words, when you screen-scrape a web site, you should be efficient in how you use computational and network resources and respectful of the owner in how you reuse the data. Consider contacting the web site owners to ask for permission.

Even though bots have negative connotations, many do recognize the positive benefits of some bots, especially search engines. If everyone were to take an extremely conservative reading of the terms of services for web sites, wouldn't many of the things we take for granted on the Internet (such as search engines) simply disappear?

Since screen-scraping web sites without public APIs is largely beyond the scope of this book, I will refer you to the following books for more information:

* *Webbots, Spiders, and Screen Scrapers* by Michael Schrenk (No Starch Press, 2007)
* *Spidering Hacks* by Kevin Hemenway and Tara Calishain (O'Reilly and Associates, 2003)

Note There's some recent research around end-user innovation that should encourage web site owners to make their sites extensible and even hackable. See Eric Von Hippel's books. Von Hippel argues that many products and innovations are originally created by users of products, not the manufacturers that then bake in those innovations after the fact (`http://en.wikipedia.org/wiki/Eric_Von_Hippel`).

---

21. Google Hacks, Third Edition by Paul Bausch, Tara Calishain, and Rael Dornfest (O'Reilly and Associates, 2006); http://proquest.safaribooksonline.com/0596527063/I_0596527063_CHP_8_SECT_8

Summary

In this chapter, I presented techniques for assessing and exploiting features of web sites that make them amenable to mashups. Specifically, you looked at web sites from the point of view of an end user. I presented a list of questions to use in analyzing web sites. Key questions include the following: What are the main resources and their URLs? How is the public being used in mashups? Does the site use tags, feeds, and weblogging features? What are the data formats for importing and exporting data? You applied these questions briefly when revisiting the mashups from the previous chapter.

The bulk of this chapter is devoted to studying URL languages of web sites and their importance in making mashups. Specifically, I presented an extensive analysis of Flickr, which has a rich URL language that covers a large part—but not all—of Flickr's functionality. I presented a simple pattern for creating that exploits the URL languages (the Mashup-by-URL-Templating-and-Embedding pattern) to create a mashup between Flickr and WordPress. I continued my examination of URL languages with a study of Google Maps, Amazon, and del.icio.us. I concluded the chapter with a discussion of screen-scraping and bots and how they can be used when public APIs are not available.

You'll turn in the next chapter to looking in depth at one group of issues raised in this chapter: tagging and folksonomies, their relationship to formal taxa, and how they can be used to knit together elements within and across sites.