CHAPTER 14

Exploring Social Bookmarking and Bibliographic Systems

One of the fundamental challenges of using the Web is keeping found things found, whether it be at the basic level of simple URLs or other digital content such as images and data sets. Social bookmarking has arisen to be a popular solution to this problem. According to "7 things you should know about social bookmarking,"[1] social bookmarking "is the practice of saving bookmarks to a public Web site and 'tagging' them with keywords." A specific type of reference is a *bookmark* or *favorite*, which is a URL stored by the user. You can try to manage these URLs in the browser by storing them as favorites or bookmarks. Social bookmarking involves storing one's references online and making bookmarking a collaborative process.

There are several reasons for using social bookmarking:

* To keep track of interesting items (URLs) you find on the Web in the hopes you will be able to find them again. This process is nicely described as "keeping found things found" by the words of a research project.[2]
* To have basic metadata or citation information about bookmarks, including metadata about the item so that you can cite the item and tell others about these items.
* To find materials that are similar to what you already have. The ability to find more and related materials is a major reason for the existence of social bookmarking. You can certainly save bookmarks in your own browser. Of course, you can just copy down the URL, paste it in a Word document, and drop the document in an e-mail message, but doing so either keeps this information to yourself or keeps it to just a select group of friends. Social bookmarks put the focus on sharing bookmarks with others so that others can find and learn from you. Tagging is widely used to forge connections among disparate sources. (Think about applying this to all we talked about in Chapter 3.)

Social bookmarking is an area in flux. At `http://www.irox.de/file_download/3` you can find a helpful chart (from 2006) comparing features of 19 systems. Wikipedia also has a list of social bookmarking sites,[3] but it's difficult to keep track of the many sites that come and go. A list that is reasonably up-to-date (last updated April 16, 2007) lists 200+ sites.[4]

Social bookmarking is of further interest in the context of this book because of the extensibility/remixability being built into these systems. Social bookmarks also lend insight into other systems. For instance, del.icio.us, the granddaddy of social

bookmarking sites, is generally credited with kicking off the latest wave of social, or *folksonomic*, tagging, which has taken many Web 2.0 sites by storm.

This chapter does the following:

* Provides an overview of the social bookmarking landscape
* Walks through a select set of social bookmarking systems: del.icio.us, Yahoo!'s MyWeb 2.0, and Connotea
* Walks through the APIs, focusing in detail on del.icio.us (which is influential and the model for other social bookmarking sites) and then comparing it to those of Yahoo! and Connotea
* Discusses how you can use the del.icio.us API to enhance Flickr through a mashup of the two web sites

1. http://www.educause.edu/ir/library/pdf/ELI7001.pdf
2. http://kftf.ischool.washington.edu/

The Social Bookmarking Scene

As mentioned, there are a lot of social bookmarking sites. A reasonable approach is to focus on del.icio.us, the one to which all other social bookmarking systems are compared. Moreover, del.icio.us has an API, and there is much to learn from it.

Here are some other social bookmarking sites I will mention and examine briefly:

* Yahoo! MyWeb 2.0 and Bookmarks because Yahoo! is pursuing these properties despite already owning del.icio.us. Functionally, Yahoo! MyWeb allows you to store pages and to use the Yahoo! identity network, a major social network.
* Connotea is a more scholarly social bookmarking site. Connotea is backed up by Nature Publishing, and therefore it's likely to have some longevity.

Using Programmableweb.com to Examine the Popularity of APIs

A good way to figure out what to focus on is to see what is listed on Programmableweb.com—and to focus on the systems that actually have APIs. Go to the following location, and look for services that are in the Bookmarks category:

```
http://www.programmableweb.com/apilist/bycat
```

As of August 11, 2007, they are as follows:

* del.icio.us
* Simpy
* Blogmarks
* Scribble
* Shadows
* Jots

* Rrove
* OnlyWire
* linkaGoGo
* Ma.gnolia

3. http://en.wikipedia.org/wiki/List_of_social_software#Social_bookmarking
4. http://3spots.blogspot.com/2006/01/all-social-that-can-bookmark.html

Table 14-1 sorts them by mashup count.[5]

*Table 14-1. A Summary of Bookmarking APIs**

API	Description	Category	Mashups
del.icio.us	Social bookmarking	Bookmarks	83
Ma.gnolia	Social bookmarking service	Bookmarks	4
Shadows	Social bookmarking and community	Bookmarks	2
Simpy	Social bookmarking	Bookmarks	1
Rrove	Social bookmarking for places; create and share maps	Bookmarks	1
Jots	Social bookmarking	Bookmarks	1

** Table generated on August 11, 2007*

The fact that del.icio.us has an order of magnitude more mashup activity than the rest of bookmark services combined is why we're focusing on del.icio.us.

del.icio.us

del.icio.us is the granddaddy of social bookmarking sites; in fact, it's the site that kicked off the whole folksonomic craze. Its web site is at `http://del.icio.us/`.

The main objects of importance in del.icio.us are bookmarks, that is, URLs. You can associate tags with a given URL. You can look at an individual's collection of URLs and the tags they use. Let's look again at the URL structures by browsing through the site and noting the corresponding URLs.

You can look at the public bookmarks for a specific user (for example, for `rdhyee`) by using this:

5. http://www.programmableweb.com/apilist/bymashups

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

You can see all the bookmarks tagged `NYTimes` by `rdhyee` by using this:

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

You can see all the URLs that people have tagged with `NYTimes` by using this:

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

And you can see just the popular ones by using this:

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

Here are today's popular items:

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

Or here are just the fresh popular ones:

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

Now, correlating a URL to a del.icio.us page is a bit trickier. Consider the following URL:

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

You can reference this from del.icio.us here:

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

How do you get `53113b15b14c90292a02c24b55c316e5` from `http://harpers.org/TheEcstasyOfInfluence.html`? Answer—it's an md5 hash. In Python the following yields `53113b15b14c90292a02c24b55c316e5`:

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

Note that the following does work:

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

It redirects to this location:

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

Using the del.icio.us API

From the documentation for the API (`http://del.icio.us/help/api/`) you can learn the following:

* All del.icio.us API calls must be sent over HTTPS.
* All calls require HTTP-Auth—that means there are no API keys *per se*, but all API calls are tied to a specific user account.
* You need to watch for the 503 HTTP error (which would mean that your calls are being throttled and that you need to slow down the rate of your calls).

Note This implies you can work on your own del.icio.us references but not on others unless they give you their credentials. All API calls to del.icio.us are made in the context of a specific user. There are no unauthenticated calls as there are on Flickr.

There are four major sections of the API:[6]

Update: Check to see when a user last posted an item.

Tags: Get a list of tags, and rename them.

Posts: Get a list of posts, add, and delete.

Bundles: Get bundles, create, and delete.

In the following sections, I'll give you a flavor of the capabilities of the del.icio.us API, but I won't comprehensively document it.

Update

You can find the documentation for the `update` method here:

```
http://del.icio.us/help/api/update
```

The `update` method tells you the last time a user updated his posts.

Let's look at three ways to work through various methods listed in the API. The first is to use a web browser, while the second and third use `curl`. Let's use the `update` method as an example:

* With a web browser, go to the following location, and when prompted, enter your del.icio.us username and password:

```
https://api.del.icio.us/v1/posts/update
```

* With `curl`, you would run the following where `USER` and `PASSWORD` are your username and password:

```
curl -u USER:PASSWORD https://api.del.icio.us/v1/posts/update
```

* Finally, you can embed the username and password into the URL:

```
curl "https://{user}:{password}@api.del.icio.us/v1/posts/update"
```

In any of these cases, if your username and password are correct, you should get a response like the following:

```
<?xml version='1.0' standalone='yes'?>
<update time="2007-04-29T22:49:55Z" />
```

6. http://del.icio.us/help/api/

Note For the following examples, we will use the second method only.

Tags

To get the complete list of tags used by a user and the number of times a given tag is used, use this:

```
curl -u USER:PASSWORD  https://api.del.icio.us/v1/tags/get
```

To rename the tag `FEDORA` to `fedora`, use this:

```
curl -u USER:PASSWORD
"https://api.del.icio.us/v1/tags/rename?old=FEDORA&new=Fedora"
```

Posts

The `posts` method has several submethods: `get`, `recent`, `all`, `dates`, and `delete`.[7]

The get Submethod

You can use the `get` submethod with the optional parameters `tag`, `dt` (for the date in `CCYY-MM-DDThh:mm:ssZ` format), and `url` to return posts matching the arguments:

```
https://api.del.icio.us/v1/posts/get?
```

For example, to get posts with the tag `mashup`, use this:

```
curl -u USER:PASSWORD https://api.del.icio.us/v1/posts/get?tag=mashup
```

You can use this submethod to figure out the number of times an article has been posted. Consider the following scenario. Say you've posted the following URL to del.icio.us:

```
http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htm
```

and want to track the number of times it has been posted to del.icio.us. You do so through this:

```
curl -u USER:PASSWORD  https://api.del.icio.us/v1/posts/get? ~CCC
url=http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htm
```

which returns the following:

```
<?xml version="1.0" standalone="yes"?>
<posts dt="2007-04-26" tag="" user="rdhyee">
  <post href="http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htm"
        description="ALA | Changing Roles of Academic and Research Libraries"
        hash="fa5be4b4401acf147ff6c8634b55cdda" others="49"
        tag="library2.0 libraries academic future" time="2007-04-26T19:45:56Z"/>
</posts>
```

7. http://del.icio.us/help/api/posts

The `others` attribute in the `post` tag gives 49, which means that 49 users have added the URL to their collection of bookmarks.

If I don't have the URL in the library, it returns this:

```
<?xml version='1.0' standalone='yes'?>
<posts dt="" tag="" user="rdhyee">
</posts>
```

You need to add a URL to your library to inquire about a given URL in the API. Another way to calculate the number of users who have a given URL in their collection of bookmarks is to note the number of `rdf:item` elements in the RSS feed for the URL. As I describe in a moment, you can access the RSS feed for a given URL here:

```
http://del.icio.us/rss/url?url={url}
```

For example:

```
http://del.icio.us/rss/url?url=http://www.ala.org/ala/acrl/acrlissues/future/~C
CC
changingroles.htm
```

The major advantages of this method are that you don't need to authenticate yourself to access the RSS feed and that you don't need to have the URL in your own set of bookmarks.

The recent Submethod

The `recent` submethod returns a list of the user's recent posts (up to 100), filtered by the optional arguments `tag` and `count`:

```
https://api.del.icio.us/v1/posts/recent?
```

For example, the following returns the last five posts:

```
curl -u USER:PASSWORD https://api.del.icio.us/v1/posts/recent?count=5
```

The all Submethod

The `all` submethod returns all your posts. You are advised to use this call sparingly (since it can generate a lot of data) and use the `update` function to see whether you need to do this call at all. You can filter by tag, such as in the following call (to get all posts with the tag `architecture`):

```
curl -u USER:PASSWORD   https://api.del.icio.us/v1/posts/all?tag=architecture
```

The add Submethod

You can use `add` to add posts to del.icio.us. It has two required parameters:

* `&url` (required) is the URL of the item.
* `&description` (required) is the description of the item.

The rest of its parameters are optional:

* `&extended` (optional) is notes for the item.
* `&tags` (optional) is tags for the item (space delimited).
* `&dt` (optional) is a date stamp of the item (with the format `CCYY-MM-DDThh:mm:ssZ`). It requires a literal `T` and `Z` as in ISO8601 at `http://www.cl.cam.ac.uk/~mgk25/iso-time.html`. This is an example: `1984-09-01T14:21:31Z`.
* `&replace=no` (optional) doesn't replace `post` if the given URL has already been posted.
* `&shared=no` (optional) makes the item private.

Let's set the description to be `ALA | Changing Roles of Academic and Research Libraries` and the tags to `library 2.0 academic ACRL technology`:

```
curl -u USER:PASSWORD "https://api.del.icio.us/v1/posts/add? ~CCC
url=http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htm~CCC
&description=ALA+%7C+Changing+Roles+of+Academic+and+Research+Libraries~CCC
&tags=library+2.0+academic+ACRL+technology"
```

This command returns this:

```
<?xml version='1.0' standalone='yes'?>
<result code="done" />
```

The dates Submethod

The `dates` submethod returns a list of dates, along with the number of posts for each date. You can optionally filter the search with a tag.

For instance, the following:

```
curl -u USER:PASSWORD https://api.del.icio.us/v1/posts/dates?tag=mashup
```

returns something like this:

```
<?xml version='1.0' standalone='yes'?>
<dates tag="mashup" user="rdhyee">
  <date count="1" date="2007-09-20" />
  <date count="1" date="2007-05-28" />
[...]
</dates>
```

The delete Submethod

To delete a post with a given URL, issue the following `GET`:

```
curl -u USER:PASSWORD "https://api.del.icio.us/v1/posts/delete? ~CCC
url={url}"
```

If the action is successful, then the result will be as follows:

```
<?xml version='1.0' standalone='yes'?>
<result code="done" />
```

Bundles

When using del.icio.us, you may quickly accumulate many tags. *Bundles* allow you to group tags into organizational sets, which you can manipulate through the del.icio.us API. I'll now illustrate how to use the API to control bundles through some examples.

The following request retrieves all the bundles for a user:

```
curl -u USER:PASSWORD https://api.del.icio.us/v1/tags/bundles/all
```

To create a bundle called `Google` to group the tags `GoogleMaps` and `GoogleEarth`, you can issue the following command:

```
curl -u USER:PASSWORD "https://api.del.icio.us/v1/tags/bundles/set? ~CCC
bundle=Google&tags=GoogleMaps+GoogleEarth"
```

You can delete the Google bundle with this:

```
curl -u USER:PASSWORD "https://api.del.icio.us/v1/tags/bundles/delete? ~CCC
bundle=Google"
```

RSS and JSON

In addition to the API, you can get RSS 1.0 feeds from del.icio.us:

http://del.icio.us/help/rss

Don't overlook them in your del.icio.us mashup work. Currently, the API returns information about the bookmarks of the authenticating user only. The RSS feeds, on the other hand, give you public information about bookmarks and how they are used by all users. Accessing RSS feeds does not require any authentication. However, you should observe the admonition to not access any given RSS feed more than once every 30 minutes.

The del.icio.us "hotlist" is at:

http://del.icio.us/rss/

The most recent postings (that have at least two posters) is here:

http://del.icio.us/rss/recent

Popular posts are here:

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

You can get recent postings by a user here:

http://del.icio.us/rss/{user}

The RSS feed for a given tag is here:

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

For example, to get the RSS feed for the tag mashup, use this:

http://del.icio.us/rss/tag/mashup

You can get a feed for posts that are tagged with both mashup and computer using this:

http://del.icio.us/rss/tag/mashup+computer

You can a feed for a specific tag and user using this:

http://del.icio.us/rss/{user}/{tag}

For example:

http://del.icio.us/rss/rdhyee/mashup+computer

Finally, you can track the history of postings for a given URL here:

http://del.icio.us/rss/url?url={url}

For example:

http://del.icio.us/rss/url?url=http://www.ala.org/ala/acrl/acrlissues/future/~CCC
changingroles.htm

You can track this feed also here:

http://del.icio.us/rss/url/fa5be4b4401acf147ff6c8634b55cdda

noting that fa5be4b4401acf147ff6c8634b55cdda is the md5 hash of http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htm.

Several RSS feeds are not in the official documentation. Posting for a user's subscription is available here:

```
http://del.icio.us/rss/subscriptions/{user}
```

A feed of a user's network (which has private information) is accessible here:

```
http://del.icio.us/rss/network/{user}?private={private-key}
```

where the `private-key` is discoverable through the del.icio.us user interface for the authenticated user. Similarly, a feed for the "links for me" feature is here:

```
http://del.icio.us/rss/for/{user}?private={private-key}
```

In addition to these RSS feeds, you can get some feeds in JSON format, which is convenient for JavaScript programming:

```
http://del.icio.us/help/json/
```

There are JSON analogs to the RSS feeds to get a user's list of posts and list of tags and details about the posting history for a given URL. Moreover, there are JSON feeds that present information about a user's social network that's not available in the RSS feeds:

* A listing of the names of people in a user's *network* at `http://del.icio.us/feeds/json/network/{user}`. That is, the list of people being tracked by the user.
* A user's *fans* at `http://del.icio.us/feeds/json/fans/{user}`. That is, the list of people tracking the user.

With these JSON feeds, you can visualize the graph of social networks in del.icio.us such as done by the tools here:

```
http://www.twoantennas.com/projects/delicious-network-explorer/
```

Third-Party Tools for del.icio.us

You can find a useful reference for what others have done with the del.icio.us API here:

```
http://del.icio.us/help/thirdpartytools
```

Of the various tools, I find useful these three useful:

* The official Firefox add-on for del.icio.us (`http://del.icio.us/help/firefox/extension`), which enables you to access your bookmarks and tags from a browser sidebar.
* MySQLicious (`http://nanovivid.com/projects/mysqlicious/`), a PHP library for copying your del.icio.us bookmarks to a MySQL database. You download the code and follow the instructions. What you end up with is a MySQL database containing all the data for your bookmarks. The documentation says PHP 4—but it works for PHP 5 in my experience.
* freshDel.icio.us (`http://freshdelicious.googlepages.com/`), a utility to check your links and prune your bookmarks.

Third-Party API Kits

Here are some of the third-party API kits listed at http://del.icio.us/help/thirdpartytools:

* PHPDelicious (http://www.ejeliot.com/pages/5).
* Pydelicious (http://code.google.com/p/pydelicious/).
* Cocoalicious (a Cocoa del.icio.us client for Mac OS X that might be a good desktop tool).
* When it comes time to mirror del.icio.us to a local database, you can look at MySQLicious for "del.icio.us to MySQL mirroring."

To give you a sense of how PHPDelicious works, the following is the code to crawl through your bookmarks and tag each bookmark with the hostname of the URL (for example, host:www.nytimes.com). Once you have such tags, you can look at all of your bookmarks from a specific domain.

```
<?php
# a file storing DELICIOUS_USER and DELICIOUS_PASSWORD
include("delicious.cred.php");
require_once('php-delicious/php-delicious.inc.php');

$del_obj = new PhpDelicious(DELICIOUS_USER, DELICIOUS_PASSWORD);

# get all your bookmarks (and check for errors in the request)
#$aPosts = $del_obj->GetAllPosts();

if (!$aPosts = $del_obj->GetAllPosts()){

  echo $del_obj->LastError(), $del_obj->LastErrorString();
  exit();

}

# go through them and extract the hostname.
# set a limit for the number of links the program does -- for debugging

$maxcount = 5;
$count = 0;

$hosts = array();

foreach ($aPosts as $post) {

  $count += 1;
  if ($count > $maxcount) {
    break;
  }

  $url = $post['url'];
  $tags = $post['tags'];
```

```
  $url_parts = parse_url($url);
  $host = $url_parts['host'];

  # make a new tag
  $host_tag = "host:" . $host;
  echo $url, " ", $host_tag, "\n";

  # add the post with the new tag
  # parameters of a post

  $sUrl = $post['url'];
  $aTags = $post['tags'];

  # add host_tag to it
  $aTags[] = $host_tag;

  # track hosts that we are seeing
  if (isset($hosts[$host_tag])) {
    $hosts[$host_tag] += 1;
  } else {
    $hosts[$host_tag] = 1;
  }

  $sDescription = $post['desc'];
  $sNotes = $post['notes'];
  $sDate = $post['updated'];
  $bReplace = true;
  echo $sUrl, $sDescription, " ", $sNotes, " ", $sDate, " ", $bReplace;
  print_r (array_unique($aTags));
  print "\n";
  if ($del_obj->AddPost($sUrl, $sDescription, $sNotes, array_unique($aTags),
$sDate,
$bReplace)) {
    print "added $sUrl successfully\n";
  } else {
    print "problem in adding $sUrl\n";
  }
}
?>
```

To give you a flavor for Pydelicious, the following is a code snippet to delete all bookmarks with a certain tag (in this example, `FlickrFavorite`):

```
USER = '[USER]'
PASSWORD = '[PASSWORD]'

import pydelicious
pyd = pydelicious.apiNew(USER,PASSWORD)

posts = pyd.posts_all(tag='FlickrFavorite')
for post in posts['posts']:
    print post['href'], "\n"
    pyd.posts_delete(post['href'])
```

Yahoo! Bookmarks and MyWeb

Yahoo!'s social bookmarking system is worth looking at because Yahoo! is a big company (with tons of users) that is adamant about getting heavily into the folksonomic space (by buying del.icio.us and Flickr, for instance). Yahoo!'s MyWeb 2.0 certainly has attractive features, including the ability to save web pages. In addition to del.icio.us, Yahoo! also has two other bookmarking services:

* `http://bookmarks.yahoo.com/`
* `http://myweb2.search.Yahoo.com/`

The relationship between the various Yahoo!-owned services is a bit confusing. The following is according to an explanation by one Yahoo! employee involved with the various bookmarking systems:[8]

* Yahoo! Bookmarks is for personal bookmarking, while del.icio.us is for social bookmarking.
* Yahoo! is extending the social bookmarking platform built for MyWeb to store the data for Yahoo! Bookmarks and soon del.icio.us. This will allow for seamless migration from one service to another while preserving existing bookmarks.
* MyWeb and Yahoo! Bookmarks share the same back-end database. In other words, they are two interfaces to the same underlying data.

Let's look at using the API documentation (`http://developer.yahoo.com/search/myweb/`). The following are the three calls currently available in the API (`yahooid=rdhyee&appid=mashupguide.net`):

* To the list of a user's (`rdhyee`) tags, use this:[9]

```
http://api.search.yahoo.com/MyWebService/V1/tagSearch? ~CCC
appid=mashupguide.net&yahooid=rdhyee&results=50
```

* To do a search for URLs with a certain tag (`mashup`), use this:

```
http://search.yahooapis.com/MyWebService/V1/urlSearch? ~CCC
appid=mashupguide.net&tag=mashup
```

* You can search for tags related to the given tag (for example, `mashup`) using this:[10]

```
http://search.yahooapis.com/MyWebService/V1/relatedTags? ~CCC
appid=mashupguide.net&tag=mashup&results=50
```

Unfortunately, there's currently no method in the API to add bookmarks to one's collection.

Connotea

Connotea is an academically oriented social bookmarking system that is run by Nature Publishing and that specializes in scientific literature:

```
http://www.connotea.org/
```

You can find the documentation for the Connotea API here:

```
http://www.connotea.org/wiki/WebAPI
```

I should distinguish between a *bookmark* and a *post* in Connotea terminology. A bookmark is a URL along with corresponding metadata, such as the title and md5 hash of the URL. A post represents an event: the adding of a bookmark to a specific user's library. Accordingly, a post contains metadata about the name of the user, the tags the user has assigned to the bookmark, and the date of the event. A bookmark may belong to many users, but a post is tied to one and only one user. You can access the bookmarks in a given user's library here:

8. http://www.techcrunch.com/2006/10/24/Yahoo!-bookmarks-enters-21st-century/#comment-297657

9. http://developer.yahoo.com/search/myweb/V1/tagSearch.html

10. http://developer.yahoo.com/search/myweb/V1/relatedTags.html

```
http://www.connotea.org/user/{user}
```

For example, you can find the bookmarks of Timo Hannay, the publishing director of Nature.com, here:

```
http://www.connotea.org/user/timo
```

There are some major conceptual similarities between the Connotea API and the del.icio.us API. For instance, both require authentication. However, in the Connotea API, you can access other users' posts.

To see whether the Connotea API recognizes your username/password combination, issue the following request:

```
curl -v -u USER:PASSWORD http://www.connotea.org/data/noop
```

Let's look next at how to get data from Connotea. You do so by forming the URL that concatenates four parts:

* The base URL of `http://www.connotea.org/data`
* An indicator of the type of data you want (bookmarks, tags, or posts), that is, `/bookmarks` or `/tags` or an empty string, which means *posts*
* Filters, any part of which is optional, specified in order by `user`, `tag`, `date`, and `uri` (that is, `/user/{username}/tag/{tagname}/date/{YYYY-MM-DD}/uri/{uri-or-hash}`)
* Optional text search parameter, number of results to return, and number to start at (that is, `?q={free-text-string}&num={number-of-results}&start={starting-index}`)

Let's look at some specific examples.

To get tags for `timo`, use this:

```
curl -u USER:PASSWORD http://www.connotea.org/data/tags/user/timo
```

In contrast to the del.icio.us API, you can get the tags of other users.

To get all the bookmarks for user `timo`, issue the following call:

```
curl -u USER:PASSWORD http://www.connotea.org/data/bookmarks/user/timo
```

To get the posts for user `timo`, issue the following call:

```
curl -u USER:PASSWORD http://www.connotea.org/data/user/timo
```

You can compare how a given URL is described as a bookmark and as a post in the two calls to see the differences between what posts and bookmarks are. A *bookmark* contains information about a given URI, its title, and which users have included (or "posted") it into their own libraries:

```
<dcterms:URI rdf:about="http://jdupuis.blogspot.com/2007/07/interview-with-timo-
~CCC
hannay-head-of-web.html">
    <link>http://jdupuis.blogspot.com/2007/07/interview-with-timo-hannay-head-
of-~CCC
web.html</link>
    <dc:title>Confessions of a Science Librarian: Interview with Timo Hannay,
Head ~CCC
of Web Publishing, Nature Publishing Group</dc:title>
    <tag>npg</tag>
    <postedBy>timo</postedBy>
    <postedBy>bk66</postedBy>
    <postedBy>andi70</postedBy>
    <postedBy>marchitelli</postedBy>
    <postedBy>darrenjones</postedBy>
    <postedBy>hjaqu001</postedBy>
    <postedBy>duncan</postedBy>
    <postedBy>bgood</postedBy>
    <postedBy>bonnieswoger</postedBy>
    <postCount>8</postCount>
    <hash>07ccdc14de0e2efee719e55c22a223b5</hash>
    <bookmarkID>1047762</bookmarkID>
    <created>2007-07-04T07:29:21Z</created>
    <updated>2007-08-14T22:00:50Z</updated>
    <firstUser>timo</firstUser>
        <citation>
          <rdf:Description>
            <citationID>465002</citationID>
            <prism:title>Interview with Timo Hannay, Head of Web Publishing,
Nature ~CCC
Publishing Group</prism:title>
            <foaf:maker>
              <foaf:Person>
                <foaf:name>John Dupuis</foaf:name>
              </foaf:Person>
            </foaf:maker>
            <dc:date>2007-07-03T00:00:00Z</dc:date>
            <journalID>433043</journalID>
            <prism:publicationName>Confessions of a Science Librarian
            </prism:publicationName>
          </rdf:Description>
        </citation>
    <rdfs:seeAlso
rdf:resource="http://www.connotea.org/data/uri/07ccdc14de0e2efee71~CCC
9e55c22a223b5" /> <!-- GET this URI to retrieve further information -->
  </dcterms:URI>
```

In addition to having some overlapping metadata, the corresponding `post` tells you when the given URL was put into the user's library:

```
<Post
rdf:about="http://www.connotea.org/user/timo/uri/07ccdc14de0e2efee719e55c22~CCC
a223b5">
  <title>Interview with Timo Hannay, Head of Web Publishing, Nature Publishing
~CCC
Group</title>
  <dc:subject>npg</dc:subject>
  <userBookmarkID>504437</userBookmarkID>
  <dc:creator>timo</dc:creator>
  <private>0</private>
  <created>2007-07-04T07:30:01Z</created>
  <updated>2007-08-09T11:33:43Z</updated>
  <uri>
    <dcterms:URI
      rdf:about="http://jdupuis.blogspot.com/2007/07/interview-with-timo-hannay-
~CCC
head-of-web.html">
      <dc:title>Confessions of a Science Librarian: Interview with Timo Hannay,
Head ~CCC
of Web Publishing, Nature Publishing Group</dc:title>
      <link>http://jdupuis.blogspot.com/2007/07/interview-with-timo-hannay-head-
of-~CCC
web.html</link>
      <hash>07ccdc14de0e2efee719e55c22a223b5</hash>
      <citation>
        <rdf:Description>
          <citationID>465002</citationID>
          <prism:title>Interview with Timo Hannay, Head of Web Publishing,
Nature ~CCC
Publishing Group</prism:title>
          <foaf:maker>
            <foaf:Person>
              <foaf:name>John Dupuis</foaf:name>
            </foaf:Person>
          </foaf:maker>
          <dc:date>2007-07-03T00:00:00Z</dc:date>
          <journalID>433043</journalID>
          <prism:publicationName>Confessions of a Science Librarian
          </prism:publicationName>
        </rdf:Description>
      </citation>
    </dcterms:URI>
  </uri>
</Post>
```

To get bookmarks that `timo` has tagged with `chemistry`, use this:

```
curl -u USER:PASSWORD
http://www.connotea.org/data/bookmarks/user/timo/tag/chemistry
```

To add a URL to your collection, do an HTTP post to this location:

```
http://www.connotea.org/data/add
```

with the mandatory parameters `uri` and `tags` and optional parameters such as `usertitle`, `description`, `myworks`, `private`, and `comment`. For instance:

```
curl -v -u USER:PASSWORD --data-binary
"uri=http://www.ala.org/ala/acrl/acrlissues/~CCC
future/changingroles.htm&tags=library2.0+academic+ACRL+technology&description=AL
A+%7~CCC
C+Changing+Roles+of+Academic+and+Research+Libraries&usertitle=Changing+Roles+of+
Acad~CCC
emic+and+Research+Libraries" http://www.connotea.org/data/add
```

To edit an existing post (for example, to change the description), use this:

```
curl -v -u USER:PASSWORD --data-binary
"uri=http://www.ala.org/ala/acrl/acrlissues/~CCC
future/changingroles.htm&tags=library2.0+academic+ACRL+technology&usertitle=Essa
y%3A~CCC
+Changing+Roles+of+Academic+and+Research+Libraries" ~CCC
http://www.connotea.org/data/edit
```

To delete the post by its URL, use this:

```
curl -v -u USER:PASSWORD --data-binary
"uri=http://www.ala.org/ala/acrl/acrlissues/~CCC
future/changingroles.htm"  http://www.connotea.org/data/remove
```

A Flickr and del.icio.us Mashup

In this section, I'll return to an idea I first wrote about in Chapter 3—demonstrating how you can create a mashup using the del.icio.us API to enhance the functionality of Flickr. There are many ways to organize the photos that you own in Flickr: you can tag them, put them in sets and collections, and send them to specific groups. With photos that belong to others, you have a lot fewer options. Generally, you're limited to making a photo a favorite. (If a photo has been placed into a group pool, you as a member of that group are able to tag the photo.) Moreover, you can't make any groupings of photos within Flickr that contain both your own photos and those of others. You can't "favorite" your own photos, and you can't place other users' photos in your sets. (You can say that group pools are an exception, but the owner of the photo has to send the photo to the pool.)

You can use del.icio.us to increase your ability to annotate Flickr photos and to intermix your own photos with those of others. A nice supporting feature of del.icio.us is that Flickr photos that are bookmarked in del.icio.us are shown with a thumbnail of the image—making deli.cio.us a simple photo display mechanism.

To use del.icio.us to track and annotate Flickr photos, you have to bookmark the photo. For bookmarking individual photos, you can simply use the same del.icio.us bookmarklets you would use to bookmark any other URL. However, if you have a large number of Flickr photos to manage with del.icio.us, it's more convenient to programmatically bookmark the photos.

Here I demonstrate how to send your Flickr favorites into del.icio.us so that you set your own tags and descriptions for the photos. The following code uses phpFlickr (see Chapter 6) and PHPDelicious:

```
<?php

# This PHP script pushes a Flickr user's favorites into a del.icio.us account

# a function for appending two input strings separated by a comma.
function aconcat ($v, $w)
{
    return $v . "," . $w;
}

# read in passwords for Flickr, the MySQL cache for phpFlickr, and del.icio.us
include("flickr_key.php");
include("mysql_cred.php");
include("delicious.cred.php");

# use phpFlickr with caching
require_once("phpFlickr/phpFlickr.php");

$api = new phpFlickr(API_KEY, API_SECRET);
$db_string =
  "mysql://" . DB_USER . ":" . DB_PASSWORD. "@" . DB_SERVER . "/". DB_NAME;
$api->enableCache(
    "db", $db_string, 10
);

# instantiate a del.icio.us object via the phpDelicious library
require_once('php-delicious/php-delicious.inc.php');
$del_obj = new PhpDelicious(DELICIOUS_USER, DELICIOUS_PASSWORD);

$username = 'Raymond Yee';

if ($user_id = $api->people_findByUsername($username)) {
  $user_id = $user_id['id'];
} else {
  print 'error on looking up $username';
  exit();
}

#print $user_id;

# get a list of the user's favorites (public ones first)
# http://www.flickr.com/services/api/flickr.favorites.getPublicList.html

# allow a maximum number of photos to be copied over -- useful for testing.
$maxcount = 2;
$count = 0;

# set the page size and the page number to start with
$per_page = 500;
$page = 1;

# loop over the pages of photos and the photos within each page

do {
```

```
if (!$photos = $api-
>favorites_getPublicList($user_id,"owner_name,last_update,tags",
$per_page, $page)) {

  echo "Problem in favorites_getPublicList call: ", $api->getErrorCode(), " ",
$api->getErrorMsg();
  exit();
}

$max_page = $photos['pages'];

foreach ($photos['photo'] as $photo) {

  $count += 1;
  if ($count > $maxcount) {
    break;
  }

  echo $photo['id'], "\n";

  # Map Flickr metadata to del.icio.us fields

  # use the URL of the context page as the del.icio.us URL
  $sUrl = "http://www.flickr.com/photos/$photo[owner]/$photo[id]/";

  # copy the photo title as the del.icio.us description
  $sDescription = $photo['title']. " (on Flickr)";

  # set del.icio.us note to empty
  $sNotes = '';

  # use the default date of now.
  $sDate = '';

  # replace previous del.icio.us posts with this URL
  $bReplace = true;

  # copy over the tags and add FlickrFavorite
  $aTags = split(' ', $photo['tags']);
  $aTags[] = 'FlickrFavorite';

  echo $sUrl, $sDescription, " ", $sNotes, " ", array_reduce($aTags, "aconcat")
,
" ", $sDate, " ", $bReplace;
  print "\n";

  if ($del_obj->AddPost($sUrl, $sDescription, $sNotes, $aTags, $sDate,
$bReplace)) {
    print "added $sUrl successfully\n";
  } else {
    print "problem in adding $sUrl\n";
  }
```

```
} // foreach

  $page += 1;

} while (($page <= $max_page) && ($count <= $maxcount)) // do

?>
```

When you run this script, you will see something akin to Figure 14-1.

Insert 858Xf1401.tif

Figure 14-1. Mashup of Flickr favorites and deli.cio.us. (Reproduced with permission of Yahoo! Inc. ® 2007 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.)

Once you have your Flickr favorites in del.icio.us, you can use all the del.ico.us functionality to manipulate them; for example, you can edit the tags (which have been copied over from Flickr), title, and description of a photo—something you couldn't do directly to the Flickr photos you don't own. In addition, you can use tags to group pictures—whether they are your own or someone else's.

These are some ways in which you can expand the functionality of the script presented here:

* You can use the del.icio.us API to help you gather your groups of Flickr photos by their del.icio.us tags and present them as slide shows.
* You can change the script to push all of a user's favorites instead of just the public favorites.
* In addition to favorites, you can add any individual photo or group of your own photos or any arbitrary Flickr aggregation of photos to del.icio.us.
* You can keep favorites synchronized between Flickr and del.icio.us.

Summary

In this chapter, you learned about social bookmarking as a whole, especially about sites that have APIs. You then concentrated on how to use the APIs of del.icio.us, Yahoo! MyWeb, and Connotea. The chapter concluded with a mashup of Flickr and del.icio.us that demonstrates how social bookmarking can be used to enhance Flickr.