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:

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:[225]

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"
  ```

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

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

```
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" />
```

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`.[226]

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?Â
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>
```

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/Â
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?Â
url=http://www.ala.org/ala/acrl/acrlissues/future/changingroles.htmÂ
&description=ALA+%7C+Changing+Roles+of+Academic+and+Research+LibrariesÂ
&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?Â
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?Â
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?Â
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/Â
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'])
```

[225] http://del.icio.us/help/api/

[226] http://del.icio.us/help/api/posts

Prev | Up | Next
Chapter 14. Exploring Social Bookmarking and Bibliographic Systems | Home | Yahoo! Bookmarks and MyWeb