Adding Delicious Data to Your Library Website

Social bookmarking services such as Delicious offer a simple way of developing lists of library resources. This paper outlines various methods of incorporating data from a Delicious account into a webpage. We begin with a description of Delicious Linkrolls and Tagrolls, the simplest but least flexible method of displaying Delicious results. We then describe three more advanced methods of manipulating Delicious data using RSS, JSON, and XML. Code samples using PHP and JavaScript are provided.

One of the primary components of Web 2.0 is social bookmarking. Social bookmarking services allow users to store bookmarks on the Web where they are available from any computer and to share these bookmarks with other users. Even better, these bookmarks can be annotated and tagged to provide multiple points of subject access.

Social bookmarking services have become popular with librarians as a means of quickly assembling lists of resources. Since anything with a URL can become a bookmark, such lists can combine diverse resource types such as webpages, scholarly articles, and library catalog records. It is often desirable for the data stored in a social bookmarking account to be displayed in the context of a library webpage. This creates consistent branding and a more professional appearance.

Figure 1. Delicious Linkroll page

Delicious (http://delicious.com/), one of the most popular social bookmarking tools, allows users to extract data from their accounts and to display this data on their own websites. Delicious offers multiple ways of doing this, from simply embedding HTML in the target webpage to interacting with the API.[1] In this paper we will begin by looking at the simplest methods for users uncomfortable with programming, and then move on to three more advanced methods using RSS, JSON, and XML. Our examples use PHP, a cross-platform scripting language that may be run on either Linux/Unix or Windows servers. While it is not possible for us to address the many environments (such as CMSes) in which websites are constructed, our code should be adaptable to most contexts. This will be especially simple in the many popular PHP–based CMSes such as Drupal, Joomla, and WordPress.

It should be noted that the process of tagging resources in Delicious requires little technical expertise, so the task of assembling lists of resources can be accomplished by any librarian. The construction of a website infrastructure (presumably by the library's webmaster) is a more complex task that may require some programming expertise.

Linkrolls and Tagrolls

The simplest way of sharing links is to point users directly to the desired

Andrew Darby (adarby@ithaca.edu) is Web Services Librarian, and **Ron Gilmour** (rgilmour@ithaca.edu) is Science Librarian at Ithaca College Library, Ithaca, New York.

Delicious page. To share all the items labeled "biology" for the user account "iclibref," one could disseminate the URL http://delicious.com/iclibref/biology. The obvious downside is that the user is no longer on your website, and they may be confused by their new location and what they are supposed to do there. Linkrolls, a utility available from the Delicious site, provides a number of options for generating code to display a set of bookmarked links, including what tags to display, the number, the type of bullet, and the sorting criterion (see figure 1).[2] This utility creates simple HTML code that can be added to a website. A related tool, Tagrolls, creates the ubiquitous Delicious tag cloud.[3]

For many librarians, this will be enough. With the embedded Linkroll code, and perhaps a bit of CSS styling, they will be satisfied with the results. However, Delicious also offers more advanced methods of interacting with data. For more control over how Delicious data appears on a website, the user must interact with Delicious through RSS, JSON or XML.

RSS

Like most Web 2.0 applications, Delicious makes its content available as RSS feeds. Feeds are available at a variety of levels, from the Delicious system as a whole down to a particular tag in a particular account. Within a library context, the most useful types of feeds will be those that point to lists of resources with a given tag. For example, the request http://feeds.delicious.com/rss/iclibref/biology returns the RSS feed for the "biology" tag of the "iclibref" account, with items listed as follows:

```
<item rdf:about="http://icarus
.ithaca.edu/cgi-bin/Pwebrecon
.cgi?BBID=237870">

  <title>Darwin's Dangerous
  Idea (Evolution 1)</title>
   <dc:date>2008-04-
  09T18:40:00Z</dc:date>
   <link>http://icarus.ithaca
  .edu/cgi-bin/Pwebrecon.
  cgi?BBID=237870</link>
   <dc:creator>iclibref</
  dc:creator>
   <description>This epi-
  sode interweaves the
  drama in key moments of
  Darwin&amp;amp;#039;s life with
  documentary sequences of
  current research, linking past
  to present and introducing
  major concepts of evolutionary
  theory. 2001</description>
   <dc:subject>biology</
  dc:subject>
   <taxo:topics>
   <rdf:Bag>
   <rdf:li rdf:resource="http://
  Delicious.com/iclibref/biol-
  ogy"/>
   </rdf:Bag>
   </taxo:topics>
   </item>
```

To display Delicious RSS results on a website, the webmaster must use some RSS parsing tool in combination with a script to display the results. The XML_RSS package provides an easy way to read RSS using PHP.[4] The code for such an operation might look like this:

```
<?php
require "XML/RSS.php"; /*
import RSS parser */
$rss = new XML_RSS("http://
del.icio.us/rss/mylibrary/biol-
ogy");
$rss->parse();
foreach ($rss->getItems() as
$item) {
echo "<div><a href='" .
$item['link'] . "'>" . $item['title']
. "</a></div>";
}
?>
```

This code uses XML_RSS to parse the RSS feed and then prints out a list of linked results.

RSS is designed primarily as a current awareness tool. Consequently, a Delicious RSS feed only returns the most recent thirty-one items. This makes sense from an RSS perspective, but it will not often meet the needs of librarians who are using Delicious as a repository of resources. Despite this limitation, the Delicious RSS feed may be useful in cases where currency is relevant, such as lists of recently acquired materials.

JSON

A second method to retrieve results from Delicious is using JavaScript Object Notation or JSON.[5] As with the RSS feed method, a request with credentials goes out to the Delicious server. The response returns in JSON format, which can then be processed using JavaScript. An example request might be http://feeds.delicious.com/v2/json/iclibref/biology. By navigating to this URL, the JSON response can be observed directly. A JSON response for a single record (formatted for readability) looks like this:

```
Delicious.posts = [

 {"u":"http:\/\/icarus.ithaca
 .edu\/cgi-bin\/Pwebrecon
 .cgi?BBID=237870",
 "d":"Darwin's Dangerous Idea
 (Evolution 1)",
 "t":["biology"],
 "dt":"2008-04-09T06:40:00Z",
 "n":"This episode interweaves
 the drama in key moments
 of Darwin's life with docu-
 mentary sequences of current
 research, linking past to present
 and introducing major concepts
 of evolutionary theory. 2001"}
 ];
```

It is instructive to look at the JSON feed because it displays the information elements that can be extracted: "u" for the URL of the resource, "d" for the title, "t" for a comma-separated list of related tags, "n" for the note field, and "dt" for the timestamp. To display results in a webpage, the feed is requested using JavaScript:

```
<script type="text/JavaScript"
src="http://feeds.Delicious.
com/v1/json/iclibref/
films?count=100"></script>
```

Then the JSON objects must be looped through and displayed as desired. Alternately, as in the script below, the JSON objects may be placed into an array for sorting.

The following is a simple example of a script that displays all of the available data with each item in its own paragraph. This script also sorts the links alphabetically.

```
 <script type="text/JavaScript"
 language="JavaScript">
var ourPosts = new Array(); /*
init ourPosts array */
/* loop through all items in the
JSON array */
for (var i=0, post; post =
Delicious.posts[i]; i++) {
ourPosts[i] = new Array(post.d,
post.u,post.n,post.t,post.dt);
}
/* Sort the resulting array
alphabetically */
ourPosts = ourPosts.sort();
/* Loop through this array, out-
putting the data in HTML */
for (var ii=0, item; item =
ourPosts[ii]; ii++) {
 document.write("<p><a
 href=\"" + ourPosts[ii][1] +
 "\">" + ourPosts[ii][0] + "</
 a>");
  if (ourPosts[ii][2] != null) {
   document.write("<br
   /><strong>Notes:</strong>
   " + ourPosts[ii][2]);
   }
  if (ourPosts[ii][3] != null) {
   document.write("<br
   /><strong>Tags:</strong> "
   + ourPosts[ii][3]);
   }
document.write("<br
/><strong>Timestamp:</
strong> " + ourPosts[ii][4] + "</
p>");
}
 </script>
```

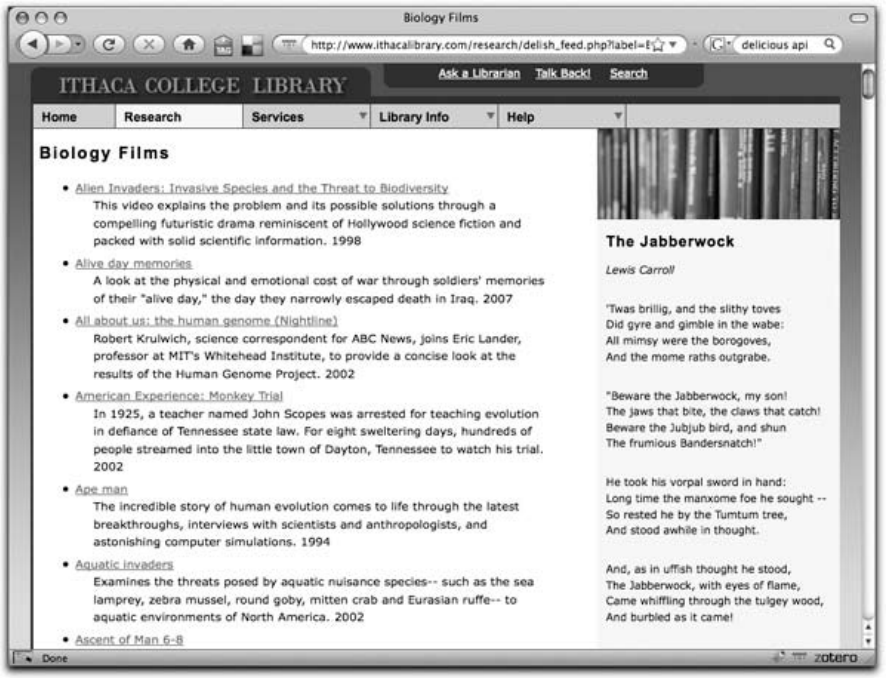

Figure 2. Biology Delicious links displayed on a library website

While RSS returns a maximum of thirty-one entries, JSON allows a maximum of one hundred. The exact number of items returned may be modified through the count parameter at the end of the URL.

At the Ithaca College Library, we chose to use JSON because at the time, Delicious did not offer the convenient Tagrolls, and the results returned by RSS were displayed in reverse chronological order and truncated at thirty-one items. Currently, we have a single PHP page that can display any Delicious result set within our library website template. Librarians generate links with parameters that designate a page title, a comma-delimited list of desired tags, and whether or not item descriptions should be displayed. For example, www.ithacalibrary.com/research/delish_feed.php?label=Biology%20Films&tag=biology,biologyI&notes=yes will return a page that looks like figure 2.

The advantage of this approach is that librarians can easily generate webpages on the fly and send the URL to their faculty members or add it to a subject guide or other webpage. The PHP script only has to read the "$_GET" variables from the URL and then query Delicious for this content.

XML

Delicious offers an application programming interface (API) that returns XML results from queries passed to Delicious through HTTPS. For instance, the request https://api.del.icio.us/v1/posts/recent?&tag=biology returns an XML document listing the fifteen most recent posts tagged as "biology" for a given account.

```
<posts tag="biology"
user="iclibref">

<post href="http://
```

```
icarus.ithaca.edu/cgi-bin/
Pwebrecon.cgi?BBID=237870"
description="Darwin's
Dangerous Idea (Evolution
1)" extended="This episode
interweaves the drama in key
moments of Darwin's life with
documentary sequences of
current research, linking past
to present and introducing
major concepts of evolutionary
theory. 2001" hash="01fc0be
0f90232caebe14157eea47d3a"
tag="biology" time="2008-04-
09T18:40:00Z"/>
<!— etc. —>
</posts>
```

Unlike either the RSS or the JSON methods, the XML API offers a means of retrieving *all* of the posts for a given tag by allowing requests such as https://api.del.icio.us/v1/posts/all?&tag=biology. This type of request is labor intensive for the Delicious server, so it is best to cache the results of such a query for future use. This involves the user writing the results of a request to a file on the server and then checking to see if such an archived file exists before issuing another request. A PHP utility called DeliciousPosts, which provides caching functionality, is available for free.[6]

Note that the username is not part of the request and must be supplied separately. Unlike the public RSS or JSON feeds, using the XML API requires users to log in to their own account. From a script, this can be accomplished using the PHP curl function:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_
URL, $queryurl);
curl_setopt($ch, CURLOPT_
USERPWD, $username . ":" .
$password);
curl_setopt($ch, CURLOPT_
RETURNTRANSFER, 1);
$posts = curl_exec($ch);
 curl_close($ch);
```

This code logs into a Delicious account, passes it a query URL, and makes the results of the query available as a string in the variable $posts. The content of $posts can then be processed as desired to create Web content. One way of doing this is to use an XSLT stylesheet to transform the results into HTML, which can then be printed to the browser:

```
/* Create a new DOM document
from your stylesheet */
$xsl = new DomDocument;
$xsl->load("mystylesheet.xsl");

/* Set up the XSLT processor */
$xp = new XsltProcessor;
$xp->importStylesheet($xsl);

/* Create another DOM docu-
ment from the contents of the
$posts variable */
$doc = new DomDocument;
$doc->loadXML($posts);

/* perform the XSLT transfor-
mation and output the resulting
HTML */
$html = $xp-
>transformToXML($doc);
echo $html;
```

Conclusion

Delicious is a great tool for quickly and easily saving bookmarks. It also offers some very simple tools such as Linkrolls and Tagrolls to add Delicious content to a website. But to exert more control over this data, the user must interact with the Delicious API or feeds. We have outlined three different ways to accomplish this: RSS is a familiar option and a good choice if the data is to be used in a feed reader, or if only the most recent items need be shown. JSON is perhaps the fastest method, but requires some basic scripting knowledge and can only display one hundred results. The XML option involves more programming but allows an unlimited number of results to be returned. All of these methods facilitate the use of Delicious data within an existing website.

References

1. Delicious, Tools, http://delicious.com/help/tools (accessed Nov. 7, 2008).

2. Linkrolls may be found from your Delicious account by clicking Settings > Linkrolls, or directly by going to http://delicious.com/help/linkrolls (accessed Nov. 7, 2008).

3. Tagrolls may be found from your Delivious account by clicking Settings > Tagrolls or directly by going to http://delicious.com/help/tagrolls (accessed Nov. 7, 2008)

4. Martin Jansen and Clay Loveless, "Pear::Package::XML_RSS," http://pear.php.net/package/XML_RSS (accessed November 7, 2008).

5. Introducing JSON, http://json.org (accessed Nov. 7, 2008).

6. Ron Gilmour, "DeliciousPosts," http://rongilmour.info/software/deliciousposts (accessed Nov. 7, 2008).

Index to Advertisers