



Use PHP to convert Twitter to RSS

Format Twitter streams as RSS with PHP libraries

Level: Intermediate

[Thomas Myer \(tom@tripleogdaremedia.com\)](mailto:tom@tripleogdaremedia.com), Consultant and Freelance Writer, Triple Dog Dare Media

03 Mar 2009

This article explains the underpinnings of Twitter and shows PHP developers how to use public libraries to extract status from their own and their friends' timelines and convert it to other formats, most notably RSS.

► **Show developerWorks content related to my search: "thomas myer"**
[site:ibm.com/developerworks](http://www.ibm.com/developerworks)

Before jumping into the topic of Twitter, I have something to get off my chest. Although I'm an avid user and fan of Twitter now, when it first came out, I was a bit nonplussed by it. In fact, I may have been a bit hostile.

There. I said it. Now let me give you a bit of back story.

When Twitter was being introduced at South by Southwest 2007, a good friend called me to tell me about it. "You should up to get an account. This is really great stuff!" Apparently, the microblogging messages were being displayed on huge monitors all across the convention hall. That was the year I was traveling a lot and couldn't actually be at South by Southwest, incidentally, I was too locked into my little world to really pay attention. Microblogging? Tweets? What the heck was this guy talking about?

A little while later, I got a chance to play around with the service and concluded that it wasn't for me. In fact, I pretty much said that Twitter was a huge waste of time. I didn't find my fellow human beings interesting enough to follow even when they invented warp drive or the cure for cancer. And, at the time, Twitter seemed to be full of folks talking about the ham sandwiches they'd just made for their kids.

Fast-forward a few months and, suddenly, I'm using Twitter all the time. I'm following news sources, presidential candidates and far-flung friends. Why the big change? I don't rightly know. Many of you fellow Twitterizens know what I'm talking about here. It's like owning your first cell phone. One day you think it's silly to walk around with an expensive phone when there are all these public pay phones, home phones, and office phones at your disposal. Then, the next thing you know, you're stranded at the side of the road in East Outersticksvillestan, and you later swear that the little thing saved your life.

One thing is for sure, though: Unlike cell phones displacing pay phones, Twitter hasn't replaced my social-networking activities — it enhances them. I normally use it to post interesting items I find on the Web — documents and resources that will make my followers think or laugh. Hopefully they're at least useful in some way. And, yes, some days all I have in me is to say, "I need coffee" or "Had a rough one last night," but I keep that sort of claptrap to a minimum.

At some point, being a developer, I want more. I want to tinker and take things apart, or at least figure out how stuff works. In a previous article, I showed you how to build a simple microblogging tool in PHP, something lightweight that would work within the context of an intranet application or other closed system.

In this article, I want to go a little deeper into Twitter's guts, exploring the API a bit, then turning to a well-known public PHP library that can facilitate some tasks. The goal of this article is to show how things work and to provide some tools with which you can export your Twitter timeline into RSS. The purpose? Sharing is the *sine qua non*, the *raison d'être* of social media services like Twitter. Sometimes it's a good thing to have your Twitter updates on your Web site — especially if, like me, you use the service to share information and resources.

Some of you may be asking, "Well, what's the point?" If you've been following the trends of social marketing and media, you

know that everyone — governments, corporate brands, marketers, consumers, enthusiasts of every stripe — is jumping on "markets are conversations" bandwagon. In fact, you could say that a large amount of digital ink is being spilled on the to

Unlike most of these people, you are a technology expert and are, therefore, well-positioned to actually know something quantifiable about these tools and the way conversations are enabled. It's good for your career — you'll be the one person in the room who actually understands all this stuff at the nuts-and-bolts level, instead of just spouting off about "social media yada yada." In addition, it's good to keep expanding your skill set.

Technical background on Twitter APIs

On the surface, Twitter is basically a cut-down blogging service. You have users who post *tweets*. Tweets are limited to 140 characters and are date- and time-stamped. Users can follow each other, which is basically a simplified syndication service. Information about Twitter accounts and associated timelines and tweets are available not only to standard Web interfaces but also to third-party applications via the Twitter API. This API exposes Twitter data and services via either a Representational State Transfer (REST) API or a search API.

This article focuses only on the REST API because you really don't need search functionality to publish an RSS feed of tweets. Needless to say, the search API is extremely useful, providing data in Atom and JavaScript Serialized Object Notation (JSON) formats, and allowing a variety of search criteria (tweets containing a word, from a particular user, to a particular user, referencing a particular user, containing a hashtag, and combinations of all the above). The REST API is equally useful providing data in RSS, XML, JSON and Atom formats.

The REST API uses HTTP basic authentication as its authentication scheme, and most libraries and utilities require the user's Twitter username and password to get started. If you don't have a Twitter account, you need one to get started or, at least, access to a Twitter account's authentication credentials. However, as the Twitter API wiki states: "All responses are relative to the context of the authenticating user." If you're trying to retrieve information from a protected user who you're not following, your request will fail.

As the name of the REST API implies, the Twitter API attempts to conform to the design principles of Representational State Transfer. REST is usually used in a loose sense to describe any simple interface that transmits domain-specific data over HTTP without any additional layers (such as SOAP or cookies). RESTful services usually orbit around one or more resources, each of which is tied to something uniquely addressable — that is, a Uniform Resource Identifier (URI). In the Twitter API context, there are various RESTful services available for extracting information about users, followers, timelines and more.

Using the Twitter API involves some familiar approaches. In many cases, you can switch from one data type to another just by changing the filename extension. Want Atom instead of RSS? Simply switch the filename extension, and, if the particular method supports it, you get a newly formatted data stream. Converting your request parameters to 8-bit Unicode Transformation Format (UTF-8) and using Uniform Resource Locator (URL) encoding is also a good idea, particularly if your requests involve complex strings.

It's important to note that the Twitter API supports the following types of requests:

- GET, which is used for data retrieval
- POST, which is used for submitting, changing, or destroying data
- DELETE, which is also used for destroying data

If you try to retrieve data with a POST operation, the API returns an error — matching methods with request types is a good place to start when debugging.

Like other REST services that use HTTP, the Twitter API sports a set of status codes and error messages. The Twitter API returns standard and appropriate HTTP status codes, described in [Table 1](#), on every request. This makes working with the Twitter API similar to (if not indistinguishable from) other HTTP-based operations you're already used to, such as Asynchronous JavaScript + XML (Ajax) and simple synchronous GET and POST operations.

Table 1. Twitter API status codes

Code	Description
------	-------------

200 OK	Everything is fine.
304 Not Modified	No new data to return.
400 Bad Request	Invalid request, with some detail. This is also what you get if you exceed the rate limit (more on this in a bit).
401 Not Authorized	You either forgot to provide authentication details, or the ones you provided are invalid.
404 Not Found	The URI you asked for doesn't exist (that is, the user doesn't exist or the data service isn't supported).
500 Internal Server Error	This is usually a problem on the Twitter side of things.
502 Bad Gateway	Twitter is down or being upgraded.
503 Service Unavailable	Twitter servers are overloaded.

Error messages are always returned in your requested format. If you request XML, you get all error messages in XML. For example, Listing 1 is a snippet of an error message published on the Twitter API wiki.

Listing 1. Snippet of an error message (XML format)

```
<?xml version="1.0" encoding="UTF-8"?>
<hash>
  <request>/direct_messages/destroy/111.xml </request>
  <error>No direct message with that ID found.</error>
</hash>
```

One final note about rate limits: Clients are allowed 100 requests per hour, starting from their first request, not including POST updates. Unauthenticated requests are tracked by IP address, while authenticated ones are tracked by the user making the request. Any time you exceed the rate limit, Twitter returns a 400 error message, so you need to think about implementing local caches on your side of things. Public timelines are cached for at least 60 seconds by Twitter, so requesting an update is pretty much a waste of time.

However, if you find yourself bumping up against the rate limit, you can request white listing from Twitter (see [Resources](#)). They get back to you within 48 hours, and, if approved, your application is allowed up to 20,000 requests per hour.

Working with the Twitter REST API

When I was just starting out as a young Web developer, I had no problem racing off in any given direction to code away night and day. Fifteen years on, I'm certainly a lot older and, hopefully, a bit wiser. Although I provide some details about some Twitter REST API methods here, it's for informational purposes only. At the end of the day, you're going to use a nice PHP library that eases your pain quite a bit.

However, because it does no good to introduce a library without some context, what follows is a brief discussion of some of the more widely used Twitter REST API methods and calls.

The most commonly used API method is probably `public_timeline`. It's available, like other status methods, in XML, JSON, RSS, and Atom formats, and is likewise retrieved using a GET method. It's available at http://twitter.com/statuses/public_timeline.xml (or `.rss`, etc.). The status element, shown in Listing 2, basically contains nodes that describe the account and has an embedded user node that describes the user.

Listing 2. The `public_timeline` status element

```
<status>
  created_at
  id
```

```
text
source
truncated
in_reply_to_status_id
in_reply_to_user_id
favorited
<user>>
  id
  name
  screen_name
  description
  location
  profile_image_url
  url
  protected
  followers_count
```

Another fairly common method is `friends_timeline`, which returns the 20 most recent statuses posted by the authenticating user and that user's friends. This is pretty much equivalent to <http://www.twitter.com/home> if you're logged to Twitter. The method is available at http://twitter.com/statuses/friends_timeline.xml (or `.rss`, `.atom`, `.json`) and is retrieved via a GET request. You can throw in some optional parameters, such as `since` (an HTTP-modified date, such as `Mon%2C+2+Feb+2009+11%3A45%3A33+GMT`) or `page` (to retrieve a specific page of tweets from the timeline).

Next, there's the `user_timeline` method, which is similar to `friends_timeline` but is limited to (you guessed it) particular user's tweets. If no Twitter username is specified, the API assumes that you want to retrieve the authenticating user's timeline.

There are a lot of other methods available, of course, but these three are enough to get you started. Now, as I alluded to, you can go about messing with the Twitter API without any help, but that doesn't make you a productive developer. You have deadlines of your own, so it's best to use a pre-built library.

Luckily for you, a fine library by the name of `twitterlibphp` is available (see [Resources](#)). Go ahead and download it, and then add it to your test or development server. In the next section, you'll start playing around with it.

Using twitterlibphp

The `twitterlibphp` library contains a list of methods that each map to a Twitter API method. The library itself handles all the connections, authentication, error handling, and the back and forth. Simply include it, authenticate, and use one of the methods to get started.

For starters, see what happens if you use the `showUser()` method to retrieve an XML status for a particular user. In Listing 3, simply replace the values for `$user` and `$pw` with your account's credentials, and that's enough to retrieve status information.

Listing 3. The `showUser()` method

```
include_once("twitter.lib.php");

$user = "your-username";
$pw = "your-password";

$twitter = new Twitter($user, $pw);

$xml = $twitter->showUser("xml", $user);

echo $xml;
```

The result provides a lot of information in its raw output. In the XML shown in Listing 4, you not only get to see my real name, screen name, and latest tweet but you can also see who I'm tweeting with, the path to my profile image, and even my color preferences.

Listing 4. Output from the `showUser()` method

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <id>14129237</id>
  <name>Thomas Myer</name>
  <screen_name>myerman</screen_name>
  <location>Most likely rushing to my next</location>
  <description>Author, Infopreneur, Blogger, Consultant</description>
  <profile_image_url>http://s3.amazonaws.com/twitter_production/profile_images/74029317/myerman_gmail.com_eef446f6_normal.jpg</profile_image_url>
  <url>http://www.triplodogs.com</url>
  <protected>>false</protected>
  <followers_count>55</followers_count>
  <profile_background_color>FFFFFF</profile_background_color>
  <profile_text_color>5F5454</profile_text_color>
  <profile_link_color>A18FB9</profile_link_color>
  <profile_sidebar_fill_color>082C35</profile_sidebar_fill_color>
  <profile_sidebar_border_color>120E26</profile_sidebar_border_color>
  <friends_count>16</friends_count>
  <created_at>Wed Mar 12 02:10:30 +0000 2008</created_at>
  <favorites_count>3</favorites_count>
  <utc_offset>-21600</utc_offset>
  <time_zone>Central Time (US & Canada)</time_zone>
  <profile_background_image_url>http://static.twitter.com/images/themes/theme1/bg.gif</profile_background_image_url>
  <profile_background_tile>>false</profile_background_tile>
  <following>>false</following>
  <notifications>>false</notifications>
  <statuses_count>427</statuses_count>
  <status>
    <created_at>Sun Feb 01 21:29:17 +0000 2009</created_at>
    <id>1167762741</id>
    <text>@adonoho [tweet snipped for privacy].</text>
    <source>web</source>
    <truncated>>false</truncated>
    <in_reply_to_status_id>1167558714</in_reply_to_status_id>
    <in_reply_to_user_id>882801</in_reply_to_user_id>
    <favorited>>false</favorited>
    <in_reply_to_screen_name>adonoho</in_reply_to_screen_name>
  </status>
</user>
```

Given all this detail, it would be easy to process this XML nodeset with SimpleXML and be able to do something useful with it all. For example, you could easily write a tool that displays the `screen_name` element along with `statuses_count` and, perhaps, the `created_at` node of the most current status.

As interesting as all this is, what you want is RSS. Specifically, you want to retrieve a user's timeline. When you have it in RSS format, you can make it available on a PHP-driven site as a sidebar widget, a unique page, or any other addressable destination.

To do that, change the previous example and use the `getUserTimeline()` method of the library, shown in Listing 5.

Listing 5. The `getUserTimeline()` method

```
include_once("twitter.lib.php");
$user = "your-username";
```

```

$pw = "your-password";

$twitter = new Twitter($user, $pw);

$rss = $twitter->getUserTimeline("rss", $user);

echo $rss;

```

As you can see, this function returns RSS ready for you to use.

Listing 6. Output from the `getUserTimeline()` method

```

<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Twitter / myerman</title>
    <link>http://twitter.com/myerman</link>
    <description>Twitter updates from Thomas Myer / myerman.</description>
    <language>en-us</language>
    <ttl>40</ttl>
  <item>
    <title>myerman: just got my copy of groundswell... also,
    just got back from seeing Taken. Liam Neeson kicks butt!</title>
    <description>myerman: just got my copy of groundswell... also, just got back from
    seeing Taken. Liam Neeson kicks butt!</description>
    <pubDate>Sun, 01 Feb 2009 04:16:55 +0000</pubDate>
    <guid>http://twitter.com/myerman/statuses/1166126064</guid>
    <link>http://twitter.com/myerman/statuses/1166126064</link>
  </item>
  <item>
    <title>myerman: For those of us trying to learn
    Objective-C - some libraries. http://tinyurl.com/dkaj4m</title>
    <description>myerman: For those of us trying to learn
    Objective-C - some libraries. http://tinyurl.com/dkaj4m</description>
    <pubDate>Fri, 30 Jan 2009 20:06:13 +0000</pubDate>
    <guid>http://twitter.com/myerman/statuses/1162686918</guid>
    <link>http://twitter.com/myerman/statuses/1162686918</link>
  </item>
  <item>
    <title>myerman: RT @andyhunter Google Chief Economist
    on innovation: We're in the middle of .. a period of "combinatorial innovation."
    http://bit.ly/xgpN</title>
    <description>myerman: RT @andyhunter Google Chief Economist
    on innovation: We're in the middle of .. a period of "combinatorial innovation."
    http://bit.ly/xgpN</description>
    <pubDate>Fri, 30 Jan 2009 15:35:18 +0000</pubDate>
    <guid>http://twitter.com/myerman/statuses/1161902779</guid>
    <link>http://twitter.com/myerman/statuses/1161902779</link>
  </item>
  <item>
    <title>myerman: A-Team movie out next year? I love it when a plan comes together.
    http://tinyurl.com/ahckx9</title>
    <description>myerman: A-Team movie out next
    year? I love it when a plan comes together.
    http://tinyurl.com/ahckx9</description>
    <pubDate>Thu, 29 Jan 2009 19:21:34 +0000</pubDate>
    <guid>http://twitter.com/myerman/statuses/1159386717</guid>
    <link>http://twitter.com/myerman/statuses/1159386717</link>
  </item>
  <item>
    <title>myerman: @andyhunter @cesart it's the pulvinar that gets you in trouble...
    every time!</title>
    <description>myerman: @andyhunter @cesart
    it's the pulvinar that gets you in trouble...
    every time!</description>
    <pubDate>Thu, 29 Jan 2009 16:44:30 +0000</pubDate>
  </item>

```

```

<guid>http://twitter.com/myerman/statuses/1158923928</guid>
<link>http://twitter.com/myerman/statuses/1158923928</link>
</item>
<item>
  <title>myerman: Create a manga avatar of yourself. http://www.faceyourmanga.com
  </title>
  <description>myerman: Create a manga avatar of yourself. http://www.faceyourmanga.com
  </description>
  <pubDate>Thu, 29 Jan 2009 16:43:55 +0000</pubDate>
  <guid>http://twitter.com/myerman/statuses/1158922223</guid>
  <link>http://twitter.com/myerman/statuses/1158922223</link>
</item>
<item>
  <title>myerman: Zombies ahead on Lamar & 15th... silly hackers!
  http://tinyurl.com/c2s3nw</title>
  <description>myerman: Zombies ahead on Lamar & 15th... silly hackers!
  http://tinyurl.com/c2s3nw</description>
  <pubDate>Thu, 29 Jan 2009 13:21:24 +0000</pubDate>
  <guid>http://twitter.com/myerman/statuses/1158382096</guid>
  <link>http://twitter.com/myerman/statuses/1158382096</link>
</item>
<item>
  <title>myerman: Obama's Mac: http://tinyurl.com/b4hsza</title>
  <description>myerman: Obama's Mac: http://tinyurl.com/b4hsza
  </description>
  <pubDate>Thu, 29 Jan 2009 13:17:58 +0000</pubDate>
  <guid>http://twitter.com/myerman/statuses/1158375012</guid>
  <link>http://twitter.com/myerman/statuses/1158375012</link>
</item>
<item>
  <title>myerman: OK, looks like I'll be
  speaking on Saturday at SxSW 2009. More details
  for the freelancer in your life: http://tinyurl.com/df8my9</title>
  <description>myerman: OK, looks like I'll be speaking on Saturday at SxSW 2009.
  More details for the freelancer in your life: http://tinyurl.com/df8my9</description>
  <pubDate>Wed, 28 Jan 2009 22:39:29 +0000</pubDate>
  <guid>http://twitter.com/myerman/statuses/1156744158</guid>
  <link>http://twitter.com/myerman/statuses/1156744158</link>
</item>
... snip
</channel>
</rss>

```

There is one caveat: You don't want to just repeatedly call this function until you hit the rate limit. Instead, you want to throttle the requests a bit. You can do something pretty clever with the `rateLimitStatus()` method and some home-rolled caching to achieve local caching effect.

The `rateLimitStatus()` method, shown in Listing 7, is a simple request that allows you to check whether you've reached your rate limit for the hour — and, no, before you ask, checking the rate-limit status doesn't count against your limit. All you have to do is supply a format and the function does the rest.

Listing 7. The `rateLimitStatus()` method

```

include_once("twitter.lib.php");

$user = "your-username";
$password = "your-pw";

$twitter = new Twitter($user, $password);
$status = $twitter->rateLimitStatus("xml");
echo $status;

```

When you run the `rateLimitStatus()` method, you get the following XML back.

Listing 8. Output from the `rateLimitStatus()` method

```
<?xml version="1.0" encoding="UTF-8"?>
<hash>
  <hourly-limit type="integer">100</hourly-limit>
  <reset-time type="datetime">2009-02-02T05:44:45+00:00</reset-time>
  <reset-time-in-seconds type="integer">1233553485</reset-time-in-seconds>
  <remaining-hits type="integer">99</remaining-hits>
</hash>
```

What you want is the value inside the `remaining-hits` node. If that ever hits five or so (or maybe 10), go to your cached RSS feed instead. With this XML nodeset available to you, all you have to do is load SimpleXML to quickly parse the rate-limit status. Of course, if you're not familiar with the vagaries of XML, it might be better to parse out JSON.

Running the same code using the JSON format yields a much more compact statement:

```
{"remaining_hits": 95, "hourly_limit": 100, "reset_time": "Mon Feb 02 05:44:45 +0000 2009",
"reset_time_in_seconds": 1233553485}
```

You could take the time to split out this string by comma and then again by colon to grab the value of `remaining_hits`. That's simple enough, of course. However, if you're running PHP V5.2.0 or higher, it's likely that you already have the `json_decode()` method available to you. As shown in Listing 9, all you have to do is use that function to pull out the `remaining_hits` number, then use a simple if statement to see if that number is anywhere close to 10 or lower. If it is, pull the RSS feed from Twitter, but save it to a file for later. If it is lower than 10, serve the saved RSS file cached locally.

Listing 9. Using the `json_decode()` method

```
include_once("twitter.lib.php");

$user = "your-username";
$pw = "your-password";

$twitter = new Twitter($user, $pw);
$status = $twitter->rateLimitStatus("json");
$fileName = $user.".rss";

$json_dump = json_decode($status);

$remaining = $json_dump->remaining_hits;

if ($remaining <= 10){
    $fh = fopen($fileName, 'r');
    $rssData = fread($fh, filesize($fileName));
    fclose($fh);
    echo $rssData;
}else{
    $rss = $twitter->getUserTimeline("rss", $user);

    //write to file
    //in case we need it!
    $fh = fopen($fileName, 'w') or die("can't write to file");
    fwrite($fh, $rss);
    fclose($fh);

    echo $rss;
}
```

```
}
```

All that's left is to give this PHP file an appropriate name (for example, `twitterfeed.php`), then use it appropriately by offering it to your users as yet another subscription option, feeding a WordPress sidebar widget, or anything like that.

Conclusion

In this article, you learned a little about how Twitter works — specifically, what the REST API exposes. You also learned how to use a publicly available PHP library to simplify your workload. With any luck, you learned enough to build your own applications and utilities.

Resources

Learn

- Wikipedia provides a great definition of [REST](#).
- Read the [Twitter REST API Documentation](#).
- [PHP.net](#) is the central resource for PHP developers.
- Check out the "[Recommended PHP reading list](#)."
- Browse all the [PHP content](#) on developerWorks.
- Follow [developerWorks on Twitter](#).
- Expand your PHP skills by checking out IBM developerWorks' [PHP project resources](#).
- To listen to interesting interviews and discussions for software developers, check out [developerWorks podcasts](#).
- Using a database with PHP? Check out the [Zend Core for IBM](#), a seamless, out-of-the-box, easy-to-install PHP development and production environment that supports IBM DB2 V9.
- Stay current with developerWorks' [Technical events and webcasts](#).
- Check out upcoming conferences, trade shows, webcasts, and other [Events](#) around the world that are of interest to open source developers.
- Visit the developerWorks [Open source zone](#) for extensive how-to information, tools, and project updates to help you develop with open source technologies and use them with IBM's products.
- Watch and learn about IBM and open source technologies and product functions with the no-cost [developerWorks demand demos](#).

Get products and technologies

- Get yourself a [Twitter](#) account.
- Sign up for [MrTweet](#) to get suggestions for whom to follow on Twitter.

- Download the [twitterlibphp](#) library by Justin Poliey.
- Request [application white listing](#) from Twitter.
- Innovate your next open source development project with [IBM trial software](#), available for download or on DVD.
- Download [IBM product evaluation versions](#), and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.
- Participate in the developerWorks [PHP Forum: Developing PHP applications with IBM Information Management products \(DB2, IDS\)](#).

About the author

Thomas Myer is the cofounder of Triple Dog Dare Media, an Austin, Texas, consulting firm. Thomas writes on the subject of knowledge management, information design, and usability. You can email him at tom@tripleddogdaremedia.com.

Share this....

 [Digg this story](#)  [del.icio.us](#)  [Slashdot it!](#)