

# Accessing the Twitter API Screencast

By Quentin Zervaas, 23 June 2010, tagged in cURL, PHP, Twitter

## Videos

- <http://www.youtube.com/watch?v=HK60FPdMBpI>

## Transcript

[0:04] In this screencast I'm going to show you how to use the Twitter API. First we'll list out all of the updates for a single user, then we'll fetch all of the updates for a list that belongs to that user.

[0:15] The API is very straightforward to use; it's just a series of REST HTTP calls, and in some cases you need to pass a username and password, and in other cases you can access the API anonymously, which we're going to do in this screencast.

[0:30] You can see the API documentation; it's at <http://apiwiki.twitter.com/Twitter-API-Documentation>, and you can see a list of functions that are available from the API.

[0:40] So I'll just click on [statuses/user\\_timeline](#). Now you can see the URL for the web service ([http://api.twitter.com/1/statuses/user\\_timeline.format](http://api.twitter.com/1/statuses/user_timeline.format)); the list of formats that are available (`xml`, `json`, `rss`, `atom`); and the different inputs.

[0:54] So let's just begin here. I'm going to paste in the URL - we'll call it `$endpoint`. I'm going to pass it to `sprintf()` because I'm going to replace a couple of parameters. Format can be `json`, because it's easy to decode. And we're going to add a parameter.

[1:32] And we'll substitute the username - we're going to use [PhpRiot](#). It doesn't say this in the documentation up the top, but down here you can see that you can insert the username just in here, so that's where we'll put that. So now we have the endpoint for this call.

[1:59] So we'll use `curl_init()` the connection. And we're going to set cURL to return the data rather than just outputting it directly, because that's of no use to us. So set `CURLOPT_RETURNTRANSFER` to `true` (using `curl_setopt()`). And that's all we need to do for setting up the connection, now we can run the request.

[2:30] So, `curl_exec()`. We're going to check the return data from the call just to make sure it was a valid request. So we'll call that `$info` and we use a `curl_getinfo()` call. And finally we close the connection (using `curl_close()`).

[2:51] So go back to the API, and if you scroll down to [Response Codes and Error Codes](#). They make good use of HTTP error codes, so if you do something like an invalid username or password you'll get the unauthorized code (401).

[3:10] To do so we check the `http_code` value from the `curl_getinfo()` call. We want to make sure it's `200`. And we add a clause here. This isn't very good error handling but it demonstrates at least what needs to be done.

[3:35] You can also check for other response codes, so as an example if you want to make sure the username and password are correct. We're not actually using username and password here so that should never happen.

[3:57] So at this stage we have our response data in `$data` and because we requested it in JSON format we can then decode it using the `json_decode()` function. We can just call this `$tweets`, and `json_decode()`, and we pass `true` to return it as an array.

[4:22] We'll just dump that out just so I can show you. I'll just `exit` and then return here and refresh. And you can see here all the data. Now there's quite a few tweets in there, so now what we need to do is make use of this data.

[4:47] The other thing to notice is this has returned a lot of data, because we haven't actually limited the results. You can see the scrollbar here indicates there's a lot of data. Let's just return back to the API call. One of the problems with the Twitter API is that some of the parameters are a bit inconsistent between calls.

[5:11] For one call it might be `per_page` whereas on this call it's `count`. So we're just going to change to `10`, just to shorten the data. That's simply a matter of adding an extra parameter. Save and return. I actually typed in `limit 1` so there's only one returned here, but if I make that `10` and refresh, now you can see there's still a lot of data but we're restricting it at least.

[5:58] Now if we return here we can remove this debug code, and now we can loop over the returned data, which is in `$tweets`, and we'll call it `$tweet`.

[6:31] We'll just give that a class, to let us style it easily. It's good practice to always use `htmlspecialchars()` so your code will be XHTML compliant. That'll escape things like ampersands. If you remember from just a second ago it was in the `text` parameter.

[6:59] So if we save that now. Now they're all on single lines here, so it's not overly useful, however you can format that as required. If you wanted to output tweets from multiple users you could output their image as well. So let's just call `$tweet['user']['profile_image_url']` and that outputs the image associated with the Twitter account.

### Listing 1. The JSON data source (timeline.php)

```
<?php
$username = 'PhpRiot';
```

# Accessing the Twitter API Screencast

By Quentin Zervaas, 23 June 2010, tagged in cURL, PHP, Twitter

```

$endpoint = sprintf(
    'http://api.twitter.com/1/statuses/user_timeline/%s.json?count=10',
    $username
);

$ch = curl_init($endpoint);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$data = curl_exec($ch);
$info = curl_getinfo($ch);

curl_close($ch);

if ($info['http_code'] == 200) {
    $tweets = json_decode($data, true);
}
else if ($info['http_code'] == 401) {
    die('Invalid credentials');
}
else {
    die('Invalid response');
}
?>
<html>
<head>
<title>PhpRiot: Twitter API</title>
</head>
<body>
<div>
<h1>PhpRiot: Twitter API - User Timeline</h1>

<?php foreach($tweets as $tweet) { ?>
<div class="tweet">

<?php echo htmlspecialchars($tweet['text']) ?>
</div>
<?php } ?>
</div>
</body>
</html>

```

[7:51] If you need a smaller image you can use CSS to resize these. So that's all there is to outputting the tweets from a single user.

[8:00] So to extend this to output tweets from multiple users, we'll change this to "List Timeline" and we're going to [list\\_statuses](#). So the URL here ([http://api.twitter.com/1/user/lists/list\\_id/statuses.format](http://api.twitter.com/1/user/lists/list_id/statuses.format)).

[8:37] So just swapping over that URL. So here is the username, and here's the list name. And as I mentioned before the limit parameters are somewhat inconsistent. This one is [per\\_page](#), and you can specify the page number as well, and specify a range, earliest and latest for which tweets are returned, but we'll just ignore this for now.

[9:03] We're going to restrict this to [30](#), and now we can specify the name of the list, which is [industry](#).

[9:23] So that's all that was required to updated the previous code. This code is identical, we don't need to change anything. The only thing you might want to change is outputting the actual tweet details.

[9:46] I'll just refresh that. So now it should show the list timeline. And as you can see here the images are a bit over the place. If we just add some CSS. And now if I refresh, these images will be a bit smaller.

## Listing 2. The main index file that displays the form (list.php)

```

<?php
$username = 'PhpRiot';
$list     = 'Industry';
$endpoint = sprintf(
    'http://api.twitter.com/1/%s/lists/%s/statuses.json?per_page=30',
    $username,
    $list
);

```

## Accessing the Twitter API Screencast

By Quentin Zervaas, 23 June 2010, tagged in cURL, PHP, Twitter

```

$ch = curl_init($endpoint);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$data = curl_exec($ch);
$info = curl_getinfo($ch);

curl_close($ch);

if ($info['http_code'] == 200) {
    $tweets = json_decode($data, true);
}
else if ($info['http_code'] == 401) {
    die('Invalid credentials');
}
else {
    die('Invalid response');
}

?>
<html>
  <head>
    <title>PhpRiot: Twitter API</title>
    <style type="text/css">
      .tweet img { width : 24px; height : 24px; }
    </style>
  </head>
  <body>
    <div>
      <h1>PhpRiot: Twitter API - List Timeline</h1>

      <?php foreach($tweets as $tweet) { ?>
        <div class="tweet">
          
          <?php echo htmlspecialchars($tweet['text']) ?>
        </div>
      <?php } ?>
    </div>
  </body>
</html>

```

[10:29] There's a lot of things you can do with this, you might want to link back to the feed of the user, so you could put in the URL here or you could auto-highlight these links. Twitter will actually do that for you if you're viewing on twitter.com but when it's in the API they don't do that for you.