Beyond HTTP GET

HTTP web services are not limited to `GET` requests. What if you want to create something new? Whenever you post a comment on a discussion forum, update your weblog, publish your status on a microblogging service like Twitter or Identi.ca, you're probably already using http `POST`.

Both Twitter and Identi.ca both offer a simple HTTP-based API for publishing and updating your status in 140 characters or less. Let's look at Identi.ca's API documentation for updating your status:

```
Identi.ca rest api Method: statuses/update
Updates the authenticating user's status. Requires the status parameter sp
ecified below. Request must be a POST.

URL
    https://identi.ca/api/statuses/update.format
Formats
    xml, json, rss, atom
HTTP Method(s)
    POST
Requires Authentication
    true
Parameters
    status. Required. The text of your status update. URL-encode as necess
ary.
```

How does this work? To publish a new message on `Identi.ca`, you need to issue an HTTP `POST` request to `http://identi.ca/api/statuses/update.format`. (The `format` bit is not part of the URL; you replace it with the data format you want the server to return in response to your request. So if you want a response in XML, you would post the request to `https://identi.ca/api/statuses/update.xml`.) The request needs to include a parameter called status, which contains the text of your status update. And the request needs to be authenticated.

Authenticated? Sure. To update your status on Identi.ca, you need to prove who you are. Identi.ca is not a wiki; only you can update your own status. Identi.ca uses HTTP Basic Authentication (a.k.a. RFC 2617) over ssl to provide secure but easy-to-use authentication. `httplib2` supports both SSL and HTTP Basic Authentication, so this part is easy.

A `POST` request is different from a `GET` request, because it includes a *payload*. The payload is the data you want to send to the server. The one piece of data that this api method *requires* is `status`, and it should be *URL-encoded*. This is a very simple serialization format that takes a set of key-value pairs (i.e. a dictionary) and transforms it into a string.

```
from urllib.parse import urlencode              #①
data = {'status': 'Test update from Python 3'}  #②
print (urlencode(data))                         #③
#'status=Test+update+from+Python+3'
```

① Python comes with a utility function to URL-encode a dictionary: `urllib.parse.urlencode()`.

② This is the sort of dictionary that the Identi.ca API is looking for. It contains one key, `status`, whose value is the text of a single status update.

③ This is what the URL-encoded string looks like. This is the *payload* that will be sent “on the wire” to the Identi.ca API server in your HTTP `POST` request.

```
from urllib.parse import urlencode
import httplib2
httplib2.debuglevel = 1
h = httplib2.Http('.cache')
data = {'status': 'Test update from Python 3'}
h.add_credentials('diveintomark', 'MY_SECRET_PASSWORD', 'identi.ca')     #①
resp, content = h.request('https://identi.ca/api/statuses/update.xml',
     'POST',                                                             #②
     urlencode(data),                                                    #③
     headers={'Content-Type': 'application/x-www-form-urlencoded'})      #④
```

① This is how `httplib2` handles authentication. Store your username and password with the `add_credentials()` method. When `httplib2` tries to issue the request, the server will respond with a `401 Unauthorized` status code, and it will list which authentication methods it supports (in the `WWW-Authenticate` header). `httplib2` will automatically construct an `Authorization` header and re-request the URL.

② The second parameter is the type of HTTP request, in this case `POST`.

③ The third parameter is the payload to send to the server. We're sending the URL-encoded dictionary with a status message.

④ Finally, we need to tell the server that the payload is URL-encoded data.

> *The third parameter to the **add_credentials()** method is the domain in which the credentials are valid. You should always specify this! If you leave out the domain and later reuse the **httplib2.Http** object on a different authenticated site, **httplib2** might end up leaking one site's username and password to the other site.*

This is what goes over the wire:

```
# continued from the previous example
send: b'POST /api/statuses/update.xml HTTP/1.1
Host: identi.ca
Accept-Encoding: identity
Content-Length: 32
content-type: application/x-www-form-urlencoded
user-agent: Python-httplib2/$Rev: 259 $

status=Test+update+from+Python+3'
reply: 'HTTP/1.1 401 Unauthorized'                          #①
send: b'POST /api/statuses/update.xml HTTP/1.1              #②
Host: identi.ca
Accept-Encoding: identity
Content-Length: 32
content-type: application/x-www-form-urlencoded
authorization: Basic SECRET_HASH_CONSTRUCTED_BY_HTTPLIB2  #③
user-agent: Python-httplib2/$Rev: 259 $

status=Test+update+from+Python+3'
reply: 'HTTP/1.1 200 OK'                                    ④
```

① After the first request, the server responds with a `401 Unauthorized` status code. `httplib2` will never send authentication headers unless the server explicitly asks for them. This is how the server asks for them.

② `httplib2` immediately turns around and requests the same URL a second time.

③ This time, it includes the username and password that you added with the `add_credentials()` method.

④ It worked!

What does the server send back after a successful request? That depends entirely on the web service API. In some protocols (like the Atom Publishing Protocol), the server sends back a `201 Created` status code and the location of the newly created resource in the Location header. Identi.ca sends back a `200 OK` and an XML document containing information about the newly created resource.

```
# continued from the previous example
print(content.decode('utf-8'))                                  #①
#<?xml version="1.0" encoding="UTF-8"?>
#<status>
# <text>Test update from Python 3</text>                        #②
# <truncated>false</truncated>
# <created_at>Wed Jun 10 03:53:46 +0000 2009</created_at>
# <in_reply_to_status_id></in_reply_to_status_id>
# <source>api</source>
# <id>5131472</id>                                              #③
# <in_reply_to_user_id></in_reply_to_user_id>
# <in_reply_to_screen_name></in_reply_to_screen_name>
# <favorited>false</favorited>
# <user>
#  <id>3212</id>
#  <name>Mark Pilgrim</name>
#  <screen_name>diveintomark</screen_name>
#  <location>27502, US</location>
#  <description>tech writer, husband, father</description>
#  <profile_image_url>http://avatar.identi.ca/3212-48-20081216000626.png</profile_image_url>
#  <url>http://diveintomark.org/</url>
#  <protected>false</protected>
#  <followers_count>329</followers_count>
#  <profile_background_color></profile_background_color>
#  <profile_text_color></profile_text_color>
#  <profile_link_color></profile_link_color>
#  <profile_sidebar_fill_color></profile_sidebar_fill_color>
#  <profile_sidebar_border_color></profile_sidebar_border_color>
#  <friends_count>2</friends_count>
#  <created_at>Wed Jul 02 22:03:58 +0000 2008</created_at>
#  <favourites_count>30768</favourites_count>
#  <utc_offset>0</utc_offset>
#  <time_zone>UTC</time_zone>
```

```
#   <profile_background_image_url></profile_background_image_url>
#   <profile_background_tile>false</profile_background_tile>
#   <statuses_count>122</statuses_count>
#   <following>false</following>
#   <notifications>false</notifications>
#</user>
#</status>
```

① Remember, the data returned by `httplib2` is always bytes, not a string. To convert it to a string, you need to decode it using the proper character encoding. Identi.ca's API always returns results in UTF-8, so that part is easy.

② There's the text of the status message we just published.

③ There's the unique identifier for the new status message. Identi.ca uses this to construct a URL for viewing the message on the web.

And here it is: