MapMyFitness API Overview

Functional Overview

This document provides everything you need in order to get started using the MapMyFitness API.

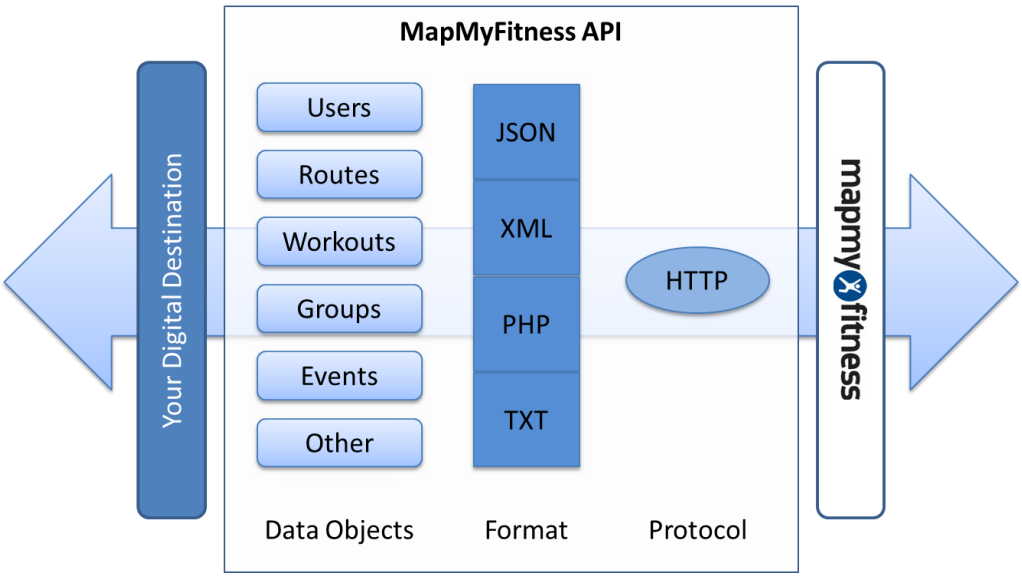
For **brands** and **partners** that want to integrate MapMyFitness (MMF) health and fitness interactions outside of the core MMF web and mobile experience, the MMF API provides developers with access to the core MMF Platform features. As a result, developers don't have to create and maintain bespoke databases and applications in order to deliver a compelling community experience that is unique to the brand and tailored to its business objectives. We've exposed the core elements of the MMF Platform, freeing architects and developers to build custom applications and other MMF-powered functionality in order to solve real-world business requirements.

For **developers** and **device manufactures** that want access to the health and fitness data across the millions of active MapMyFitness community members, the MMF API allows device and app developers to instantly add a compelling and social digital experience by publishing member activity to our community. By plugging into the MapMyFitness community, your app or device instantly becomes part of a fully engaging experience where members can track and share activities, join groups and participate in challenges.

Technical Overview

The MapMyFitness API is a web-services style API that allows you to interact with the core objects within the MapMyFitness Platform. Interaction can be simple queries or more advanced permission-based create / edit / delete functionality on selected objects.

All API methods are available for use via the HTTP interface, and data can be returned in XML, JSON, PHP or text. Some methods require user authentication which can be achieved using OAuth industry standards.



Object Model & Methods

The MMF primary content objects include the following:

- Users
- Routes
- Workouts
- Activity Feed
- Groups
- Events
- Gear
- OAuth

Getting Started

You can be up and running in 4 simple steps:

Step 1: Review the MMF API documentation and MMF OAuth Tutorial to get a basic understanding of what functionality is supported by the API and so that you can plan your application or device integration strategy. The MMF API documentation can be found here: http://api.mapmyfitness.com/3.1/

Step 2: Review and accept the MMF API Terms of User by clicking 'Get Access' here: http://www.mapmyrun.com/developer/

Step 3: Request a MMF API key here: http://www.mapmyrun.com/developer/api_keys/add/

Step 4: Keep us posted on your most creative and compelling integrations so that we can share with the broader MMF community!

MapMyFitness API OAuth 1 Tutorial

Confidential & Proprietary

The following is a complete end-to-end tutorial that describes how to use OAuth with the MapMyFitness API. The API uses OAuth 1, as specified in RFC 5849.

In this tutorial, you will:

1. Register a new client application.
2. Write and execute a simple client application.

Your client application will:

1. Use its "client credentials" to obtain "temporary credentials".
2. Prompt a user to authorize the client application using the "temporary credentials token".
3. Receive the user's authorization "verifier".
4. Use the user's authorization "verifier" and "temporary credentials" to obtain "token credentials".
5. Use the "token credentials" to request an API resource.

Note:

> Since "RFC 5849" is such a mouthful, we refer to the OAuth used by the MapMyFitness API as "OAuth 1" to distinguish it from the newer, different OAuth 2. As of 2013-01-17, OAuth 1 and OAuth 2 are both current specifications, and they differ in the way they specify that clients and servers should negotiate access on a user's behalf.
>
> Confusingly, there are three different OAuth specifications that use "1" in their name -- OAuth 1.0, OAuth 1.0a, and RFC 5849: The OAuth 1.0

> Protocol. More confusingly, in content, RFC 5849 is actually equivalent to 1.0a, not 1.0 as its title would suggest. As of 2013-01-17, RFC 5849 is the current OAuth 1 specification, and the MapMyFitness API aims to conform to it.

A word of caution about OAuth 1 request signing

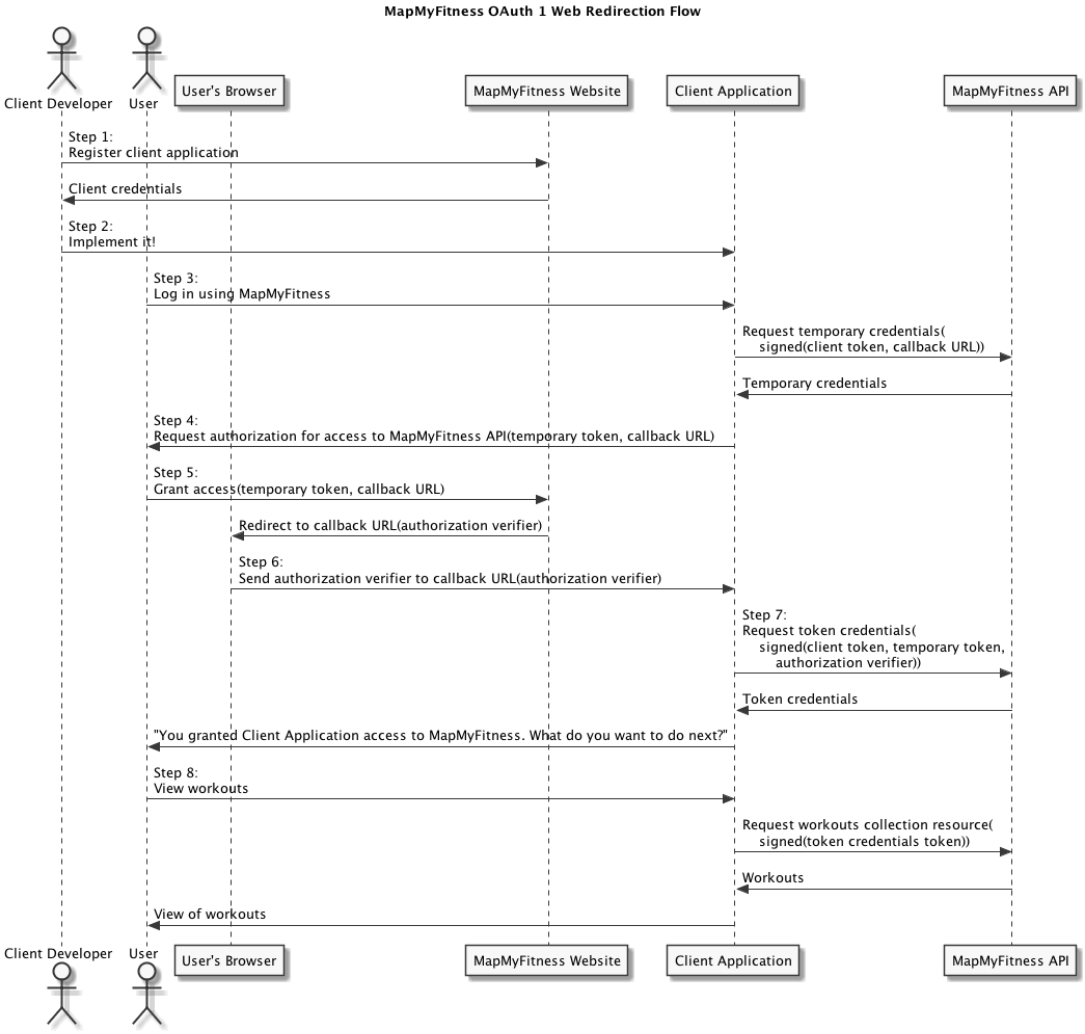
OAuth 1 uses request signing to avoid sending shared secrets between the client and server any more than necessary. It's easy to get request signing wrong, so MapMyFitness recommends using an established OAuth 1 client library rather than implementing the specification yourself.

This tutorial demonstrates the process using Python's Requests, requests-oauthlib, and oauthlib packages, which are freely available, so that you can reproduce the process step for step before attempting to implement it in your own programming environment.

More background: The actors and how they communicate

The complete OAuth process involves six different participants collaborating to facilitate access by your client application to a user's data stored at MapMyFitness.

The following diagram shows the sequence of interactions among the participants.



- Client Developer (You!)

  A MapMyFitness user who registers the client application.

- Client Application

  The application you design and implement.

- User

  A MapMyFitness user who uses your Client Application.

- User's Browser

The web browser that the User uses to interact with the MapMyFitness Website and your Client Application.

- MapMyFitness website

  User interfaces for you, the Client Developer, to register your client application and manage info about it; and user interfaces for the end User to manage the access he grants to client applications like yours.

- MapMyFitness API server

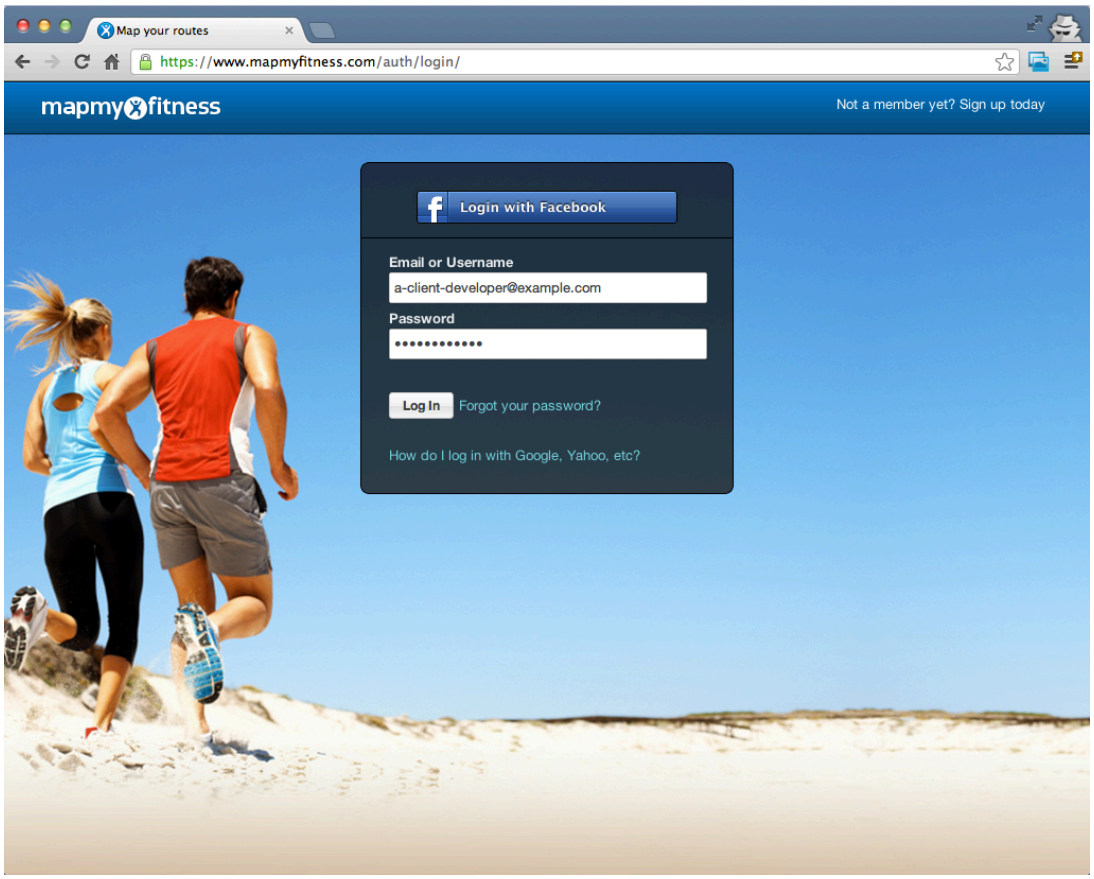
  The programming platform that your Client Application uses to negotiate access to MapMyFitness resources and manipulate those resources on behalf of a User.

This tutorial will guide you through the steps labeled in the diagram.

Tutorial

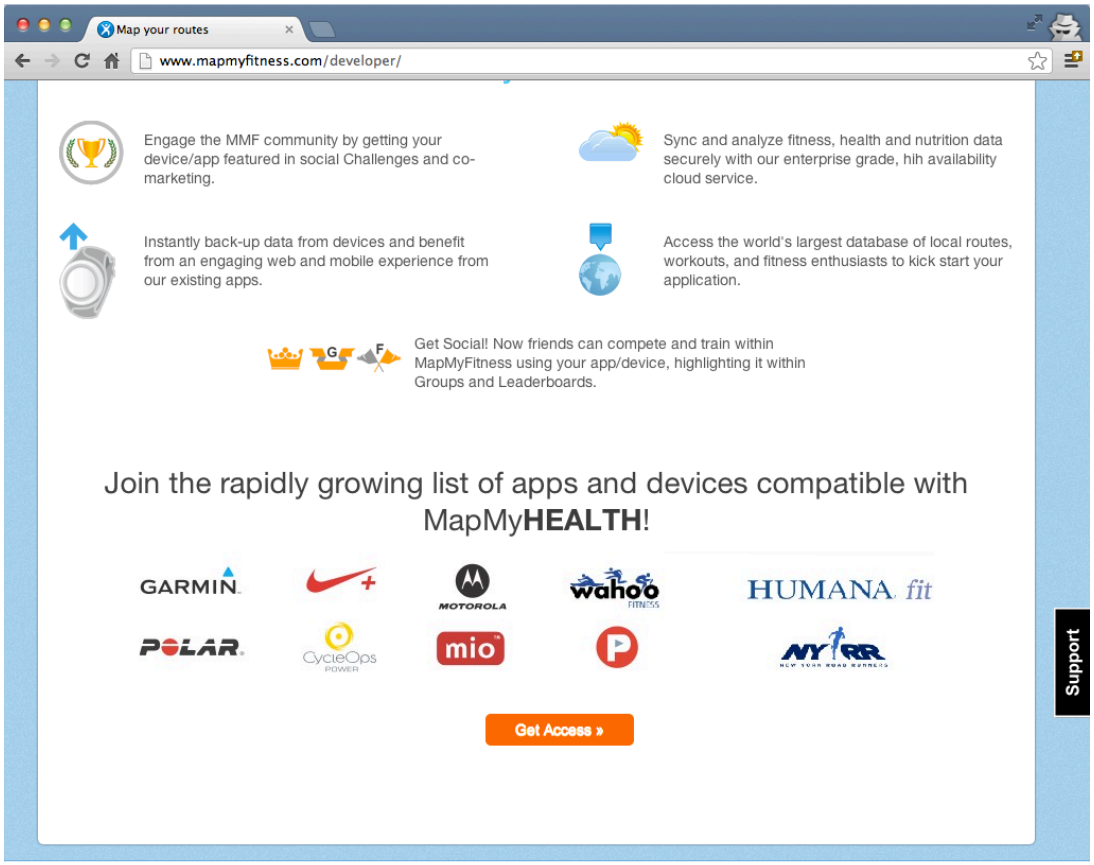
Step 1: Client developer: Register a client application

Log into the MapMyFitness website.



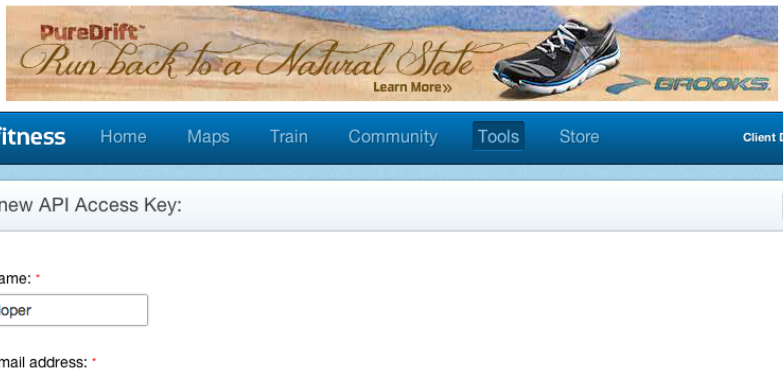
Visit http://www.mapmyfitness.com/developer.

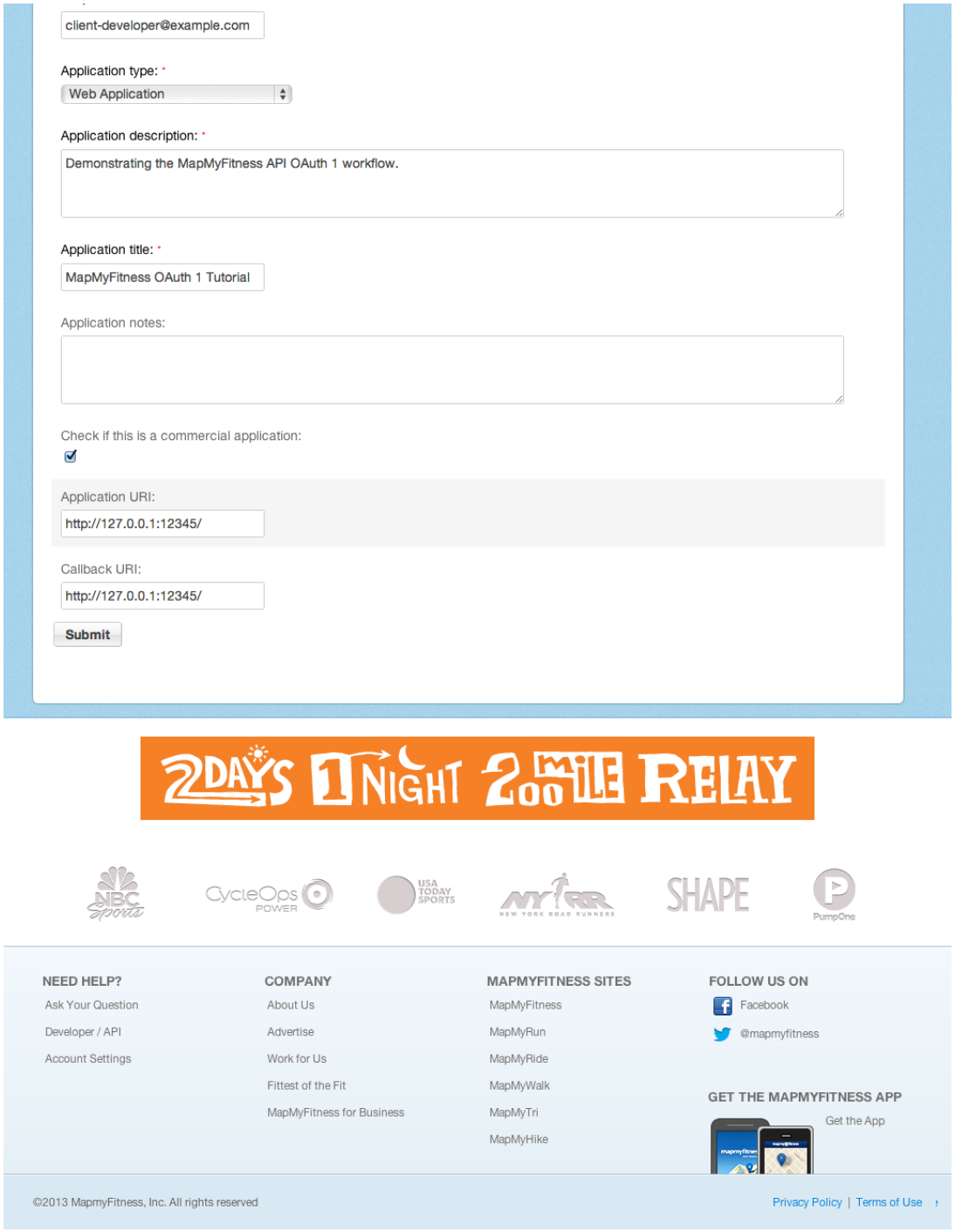
Scroll down and click Get Access.



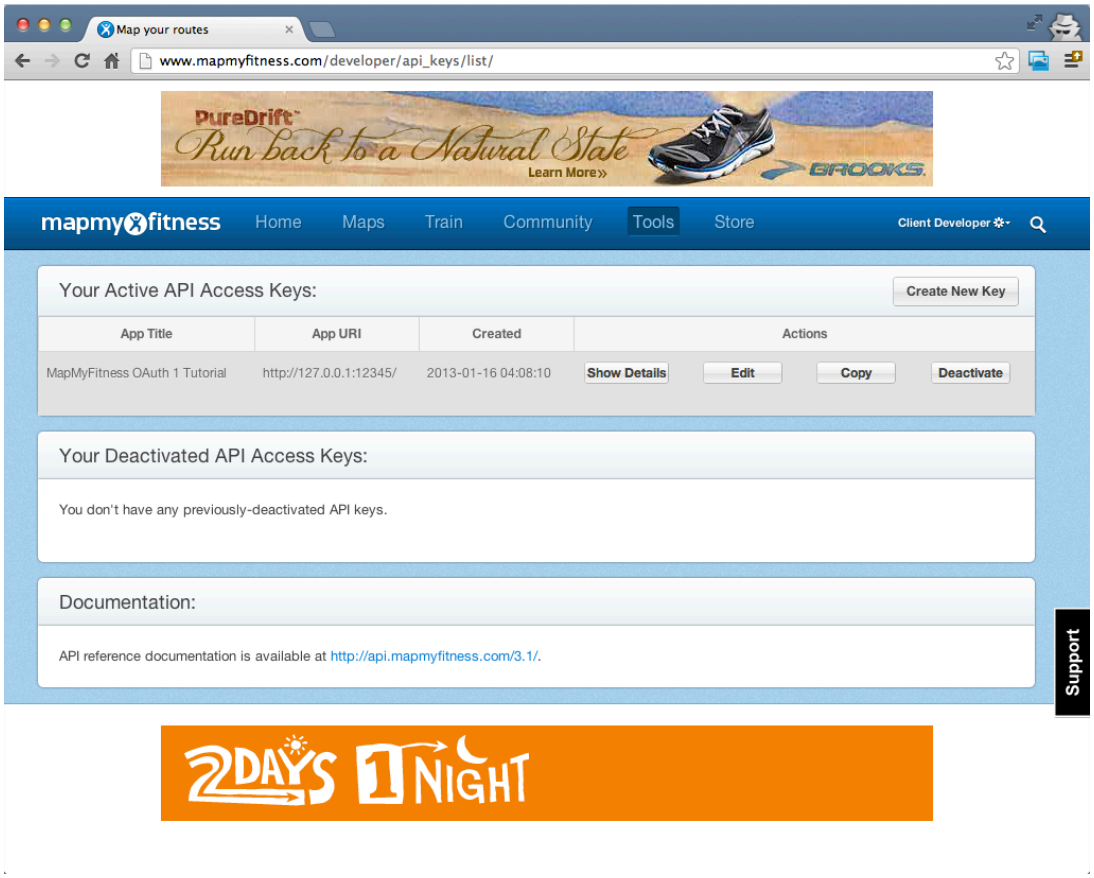
Complete the form with information about your client application.

The Callback URI is a link to which the API will redirect the user's browser after the user authorizes your application. The request to the Callback URI will include information your client application needs to obtain token credentials. Later, the tutorial will show how to extract that information in a web server handling the request.

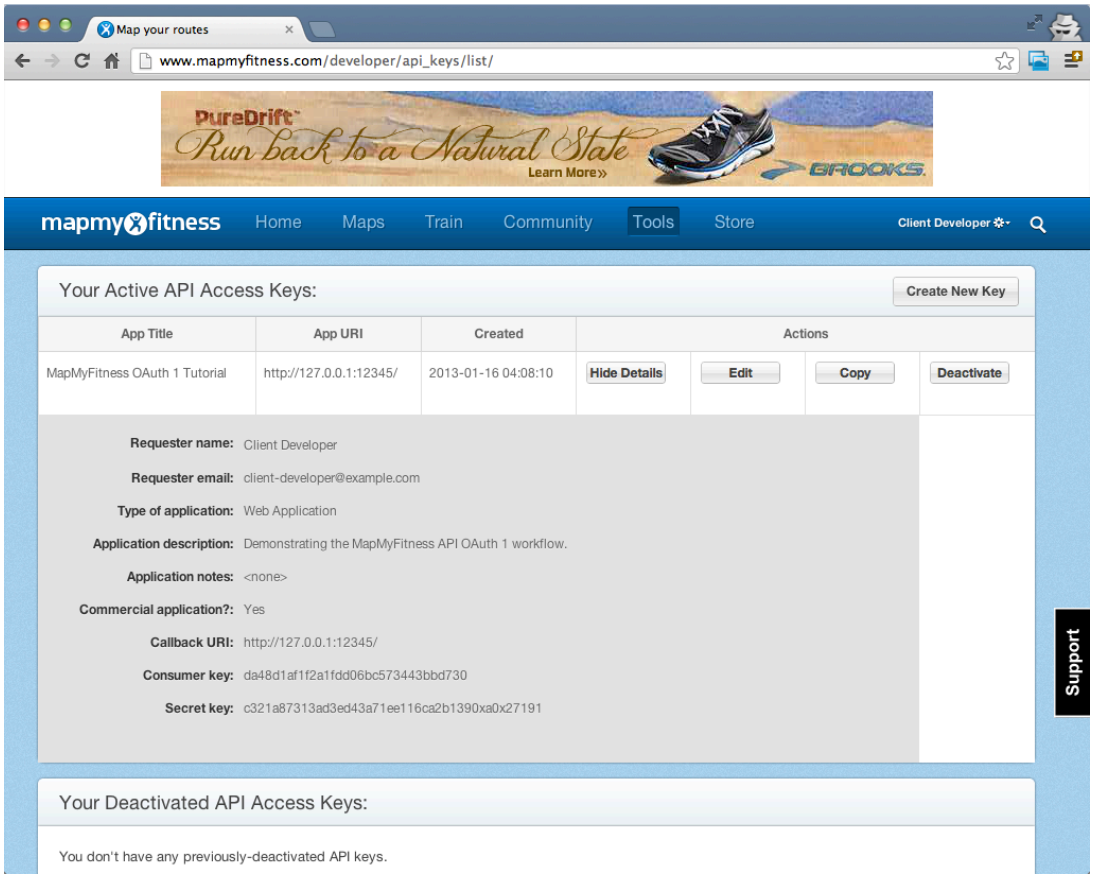




When you submit the form, you should see your registered client application in a list. Click Show Details.



Make a note of your new application's Consumer Key (client token) and Secret Key (client token secret). You should keep your Secret Key private in a secure location.



Log out of the MapMyFitness website. Later, you'll act as a user of the tutorial client application and log into the MapMyFitness website to authorize the client application's access to your MapMyFitness account.

Step 2: Client developer: Get the software to implement the tutorial client application.

The tutorial assumes you have:

- Python >= 2.6
- pip
- virtualenv

If not, download Python from the Python website and install it.

Install setuptools.

Install pip.

```
[client-dev@dev-machine:~]
$ sudo easy_install pip
```

Install virtualenv.

```
[client-dev@dev-machine:~]
$ sudo pip install virtualenv
```

Create a virtualenv and activate it.

```
[client-dev@dev-machine:~]
$ mkdir mmf-api-oauth-1-tutorial
[client-dev@dev-machine:~]
$ cd mmf-api-oauth-1-tutorial
[client-dev@dev-machine:~/mmf-api-oauth-1-tutorial]
$ virtualenv env
New python executable in env/bin/python
Installing setuptools.............done.
Installing pip...............done.
[client-dev@dev-machine:~/mmf-api-oauth-1-tutorial]
$ . env/bin/activate
```

Install IPython, Requests, requests-oauthlib, and oauthlib.

```
(env)[client-dev@dev-machine:~/mmf-api-oauth-1-tutorial]
$ pip install ipython
Downloading/unpacking ipython
  Downloading ipython-0.13.1.tar.gz (5.9MB): 5.9MB downloaded
  Running setup.py egg_info for package ipython
    running egg_info

Installing collected packages: ipython
  Running setup.py install for ipython
    running install

    Installing ipcontroller script to /Users/mlm/mmf-api-oauth
```

```
-1-tutorial/env/bin
    Installing iptest script to /Users/mlm/mmf-api-oauth-1-tut
orial/env/bin
    Installing ipcluster script to /Users/mlm/mmf-api-oauth-1-
tutorial/env/bin
    Installing ipython script to /Users/mlm/mmf-api-oauth-1-tu
torial/env/bin
    Installing pycolor script to /Users/mlm/mmf-api-oauth-1-tu
torial/env/bin
    Installing iplogger script to /Users/mlm/mmf-api-oauth-1-t
utorial/env/bin
    Installing irunner script to /Users/mlm/mmf-api-oauth-1-tu
torial/env/bin
    Installing ipengine script to /Users/mlm/mmf-api-oauth-1-t
utorial/env/bin
Successfully installed ipython
Cleaning up...
(env)[client-dev@dev-machine:~/mmf-api-oauth-1-tutorial]
$ pip install git+https://github.com/matthewlmcclure/requests.
git@628e393b9a251c3d2b9910c697b9e4ac4f3d8d6a#egg=requests git+
https://github.com/matthewlmcclure/requests-oauthlib.git@66021
a8b323881519761d4fbab9491f0f0f9a12e#egg=requests-oauthlib oaut
hlib==0.3.4
Downloading/unpacking requests from git+https://github.com/mat
thewlmcclure/requests.git@628e393b9a251c3d2b9910c697b9e4ac4f3d
8d6a#egg=requests
  Cloning https://github.com/matthewlmcclure/requests.git (to
628e393b9a251c3d2b9910c697b9e4ac4f3d8d6a) to ./env/build/reque
sts
  Could not find a tag or branch '628e393b9a251c3d2b9910c697b9
e4ac4f3d8d6a', assuming commit.
  Running setup.py egg_info for package requests

Downloading/unpacking requests-oauthlib from git+https://githu
b.com/matthewlmcclure/requests-oauthlib.git@66021a8b3238815197
61d4fbab9491f0f0f9a12e#egg=requests-oauthlib
  Cloning https://github.com/matthewlmcclure/requests-oauthlib
.git (to 66021a8b323881519761d4fbab9491f0f0f9a12e) to ./env/bu
ild/requests-oauthlib
  Could not find a tag or branch '66021a8b323881519761d4fbab94
```

```
91f0f0f9a12e', assuming commit.
  Running setup.py egg_info for package requests-oauthlib

Downloading/unpacking oauthlib==0.3.4
  Downloading oauthlib-0.3.4.tar.gz (54kB): 54kB downloaded
  Running setup.py egg_info for package oauthlib

Installing collected packages: requests, requests-oauthlib, oa
uthlib
  Running setup.py install for requests

  Running setup.py install for requests-oauthlib

  Running setup.py install for oauthlib

Successfully installed requests requests-oauthlib oauthlib
Cleaning up...
```

Note:

> Install the specific versions above. As of 2013-01-11, Requests is undergoing a major refactoring that has led to some defects in the default versions.

Step 3: Client application: Obtain temporary credentials

We'll implement a client application in an interactive Python shell to demonstrate exactly what's going on. In a real application, this is the kind of process that your web application server would implement so that your server can make MapMyFitness API requests.

Start the IPython interactive shell.

```
(env)[client-dev@dev-machine:~/mmf-api-oauth-1-tutorial]
$ ipython
Python 2.7.3 (default, Sep 15 2012, 19:45:36)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra
details.

In [1]:
```

Import the packages we need.

```
In [1]: from __future__ import unicode_literals

In [2]: import requests

In [3]: from requests_oauthlib import OAuth1
```

Copy, paste your application's client token and client token secret from the application details on the website in Step 1.

```
In [4]: client_key = 'da48d1af1f2a1fdd06bc573443bbd730'

In [5]: client_secret = 'c321a87313ad3ed43a71ee116ca2b1390xa0x
27191'
```

Configure the MapMyFitness API URL to obtain temporary credentials.

```
In [6]: temporary_credentials_url = 'http://api.mapmyfitness.c
om/3.1/oauth/request_token'
```

Configure requests-oauthlib to sign requests for our client application. To obtain access tokens, your client application's "web redirection flow" requests will use

the Form-Encoded Body method to send OAuth parameters, as described in RFC 5849.

```
In [7]: callback_uri='http://127.0.0.1:12345/'

In [8]: oauth = OAuth1(client_key, client_secret=client_secret
, callback_uri=callback_uri, signature_type='BODY')
```

Make an API HTTP request to obtain temporary credentials.

```
In [9]: r = requests.post(url=temporary_credentials_url, heade
rs={'Content-Type': 'application/x-www-form-urlencoded', 'Acce
pt': 'application/x-www-form-urlencoded'}, auth=oauth)
```

Let's see what the request body looked like.

```
In [10]: r.request.body
Out[10]: u'oauth_nonce=61346384004511306311358332274&oauth_tim
estamp=1358332274&oauth_version=1.0&oauth_signature_method=HMA
C-SHA1&oauth_consumer_key=da48d1af1f2a1fdd06bc573443bbd730&oau
th_callback=http%3A%2F%2F127.0.0.1%3A12345%2F&oauth_signature=
ISQWD4GyTDVBQY6GPiu6q4641jM%3D'
```

Parse the temporary credentials from the response body.

```
In [11]: r.content
Out[11]: 'oauth_token=46eac8f31bce6ff37dd408976e5cabe4050f6b28
5&oauth_token_secret=0fba5a666475d0709e28ca714f7a05d5'

In [12]: import urlparse

In [13]: temporary_credentials = urlparse.parse_qs(r.content)

In [14]: temporary_credentials
Out[14]:
{'oauth_token': ['46eac8f31bce6ff37dd408976e5cabe4050f6b285'],
 'oauth_token_secret': ['0fba5a666475d0709e28ca714f7a05d5']}
```

Step 4: Client application: Prompt a user to authorize the client application

Add `oauth_token` and `oauth_callback` query parameters to the authorization URL.

```
In [15]: from oauthlib.oauth1.rfc5849.utils import escape

In [16]: authorize_url = 'https://www.mapmyfitness.com/oauth/a
uthorize/?oauth_token=%s&oauth_callback=%s' % (temporary_crede
ntials['oauth_token'][0], escape(callback_uri))

In [17]: authorize_url
Out[17]: u'https://www.mapmyfitness.com/oauth/authorize/?oauth
_token=46eac8f31bce6ff37dd408976e5cabe4050f6b285&oauth_callbac
k=http%3A%2F%2F127.0.0.1%3A12345%2F'
```

Direct the user's browser to the authorization URL.

```
In [18]: import webbrowser

In [19]: webbrowser.open(authorize_url)
Out[19]: True
```

Make your client application server listen for the callback request from the user's forthcoming authorization verification request.

```
In [20]: from BaseHTTPServer import HTTPServer, BaseHTTPReques
tHandler

In [21]: class AuthorizationHandler(BaseHTTPRequestHandler):
   ....:
   ....:     def do_GET(self):
   ....:         self.send_response(200, 'OK')
   ....:         self.send_header('Content-Type', 'text/html')
   ....:         self.end_headers()
   ....:         self.server.path = self.path
   ....:

In [22]: server_address = (urlparse.urlparse(callback_uri).hos
tname, urlparse.urlparse(callback_uri).port)

In [23]: httpd = HTTPServer(server_address, AuthorizationHandl
er)

In [24]: httpd.handle_request()
```

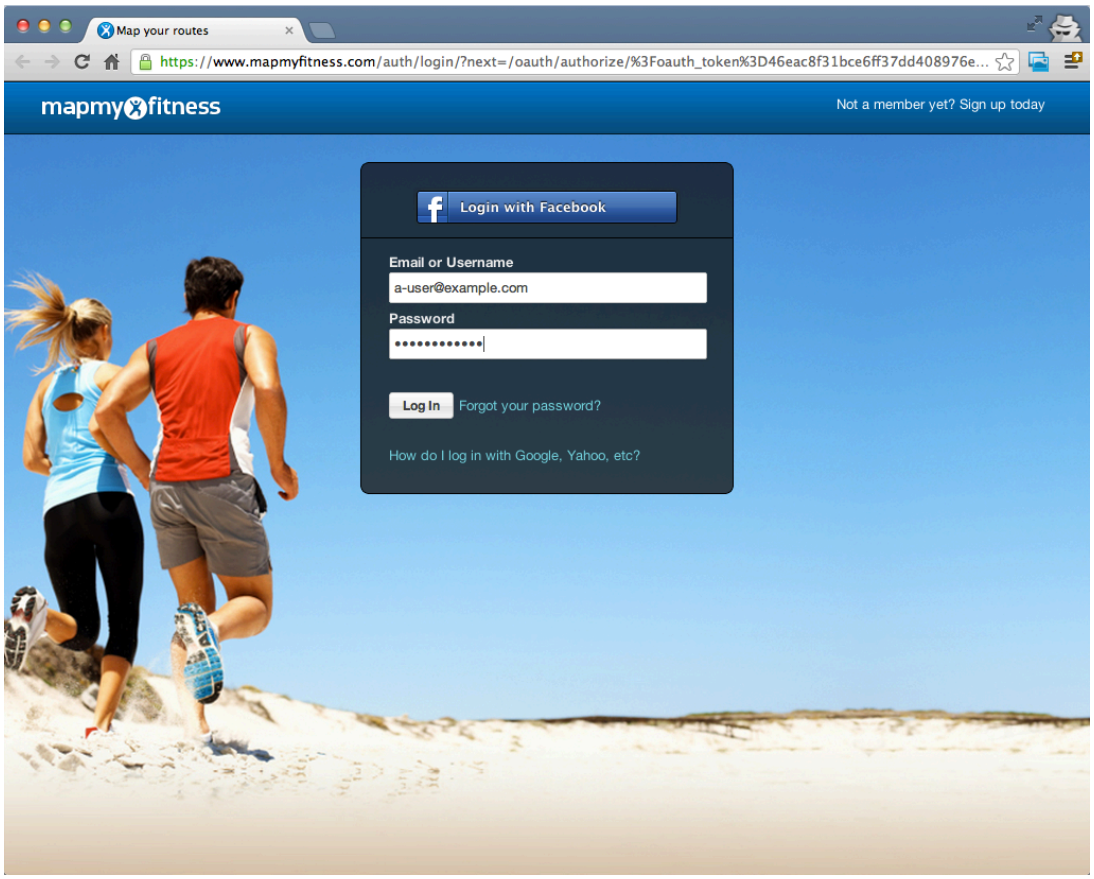
The `httpd.handle_request()` statement will block until the user's browser makes the authorization verification callback request to your client application server.

Step 5: User: Authorize the client application to access the MapMyFitness API on your behalf

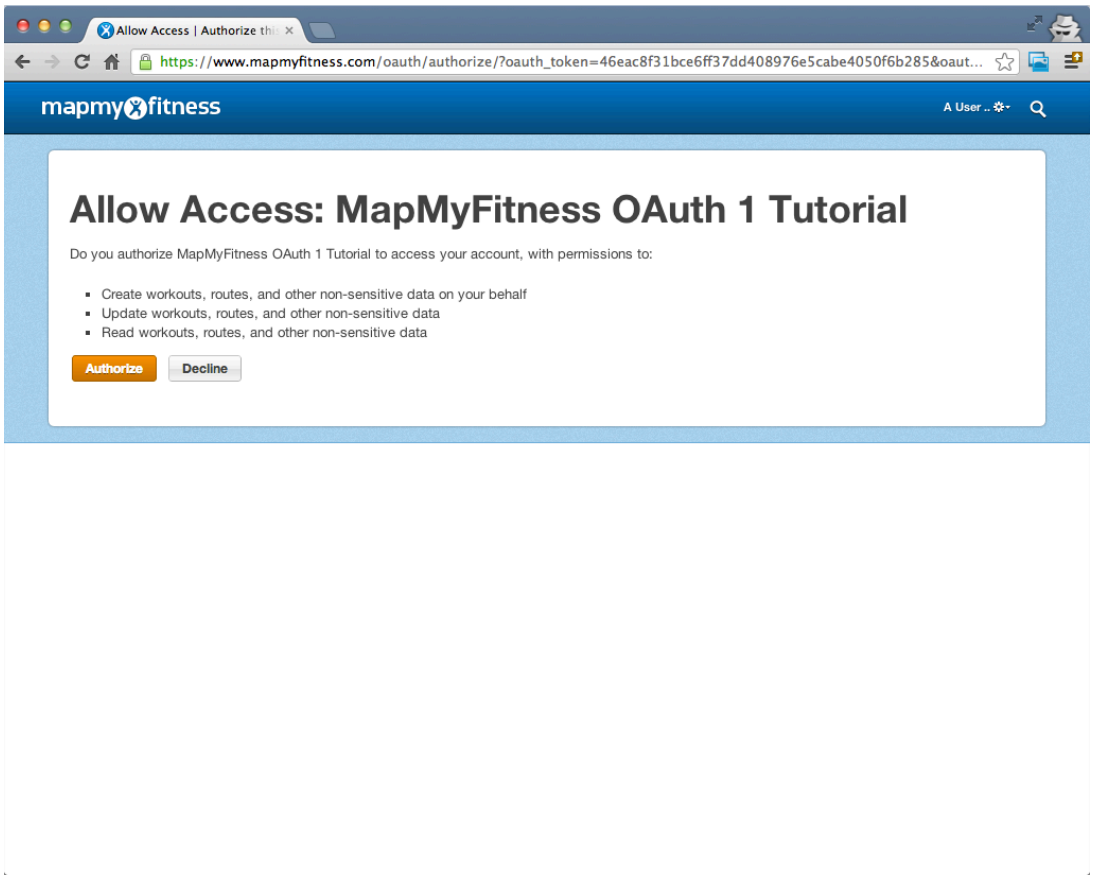
In this step, you'll act as a user of the tutorial client application. In the web page that the client application opened when it called `webbrowser.open()` earlier, enter your username and password, and click Log In.

Note:

> To simplify, log in using your client developer account. But remember, in this step, you're acting as a user of your client application, not as its developer.



Click Authorize.



The tutorial' client application web server returns an empty page when `httpd.handle_request()` returns. In a real client application, `httpd.handle_request()` would do the work in the subsequent tutorial steps, and *then* return a response page with actions that the user could take next.

Step 6: Client application: Receive the user's authorization verifier

When the user clicks Authorize, his browser makes a request to the MapMyFitness website server. The website server confirms the user's authorization and responds to the user's browser with a redirect to your client application's callback URL. The user's browser follows the redirect and makes a request to your client application server's callback URL including the authorization verifier.

The `httpd.handle_request()` statement that blocked in Step 4 returns when it handles the request from the user's browser.

```
In [25]: httpd.handle_request()
1.0.0.127.in-addr.arpa - - [16/Jan/2013 09:03:34] "GET /?oauth
_token=46eac8f31bce6ff37dd408976e5cabe4050f6b285&oauth_verifie
r=5891 HTTP/1.1" 200 -
```

Parse the temporary credentials token and verifier from the request query string.

```
In [26]: verifier_url = urlparse.urlparse(httpd.path)

In [27]: verifier_url
Out[27]: ParseResult(scheme='', netloc='', path='/', params=''
, query='oauth_token=46eac8f31bce6ff37dd408976e5cabe4050f6b285
&oauth_verifier=5891', fragment='')

In [28]: verifier_query = urlparse.parse_qs(verifier_url.query
)

In [29]: verifier_query
Out[29]:
{'oauth_token': ['46eac8f31bce6ff37dd408976e5cabe4050f6b285'],
 'oauth_verifier': ['5891']}
```

Step 7: Client application: Exchange the temporary credentials and authorization verifier for token credentials

Let's look at the temporary credentials again.

```
In [30]: temporary_credentials
Out[30]:
{'oauth_token': ['46eac8f31bce6ff37dd408976e5cabe4050f6b285'],
 'oauth_token_secret': ['0fba5a666475d0709e28ca714f7a05d5']}
```

Add all of the following to the OAuth parameters:

- client token
- temporary token
- callback URI
- authorization verifier

And use two secrets to sign the request:

- client token secret
- temporary token secret

```
In [31]: oauth = OAuth1(client_key, client_secret, unicode(tem
porary_credentials['oauth_token'][0]), unicode(temporary_crede
ntials['oauth_token_secret'][0]), callback_uri='http://127.0.0
.1:12345/', signature_type='BODY', verifier=verifier_query['oa
uth_verifier'][0])

In [32]: token_credentials_url = 'http://api.mapmyfitness.com/
3.1/oauth/access_token'

In [33]: r = requests.post(url=token_credentials_url, headers=
{'Content-Type': 'application/x-www-form-urlencoded', 'Accept'
: 'application/x-www-form-urlencoded'}, auth=oauth)
```

Parse the token credentials from the response body.

```
In [34]: r.content
Out[34]: 'oauth_token=fdd6225e87c896be8abf2fb54974c98a050f6b50
8&oauth_token_secret=99a4f928ee10adaf20f816f4ce85e83e'

In [35]: token_credentials = urlparse.parse_qs(r.content)

In [36]: token_credentials
Out[36]:
{'oauth_token': ['fdd6225e87c896be8abf2fb54974c98a050f6b508'],
 'oauth_token_secret': ['99a4f928ee10adaf20f816f4ce85e83e']}
```

Notice that the token credentials are different from the temporary credentials.

```
In [37]: temporary_credentials
Out[37]:
{'oauth_token': ['46eac8f31bce6ff37dd408976e5cabe4050f6b285'],
 'oauth_token_secret': ['0fba5a666475d0709e28ca714f7a05d5']}
```

Step 8: Client application: Request a resource on behalf of the user with his token credentials

It's helpful to think of OAuth 1 as describing two separate specifications. At this point, you've completed a demonstration of the first: the web redirection flow for obtaining a user's authorization and corresponding API token credentials.

Once your application gets this far, it can use the token credentials to make resource requests on behalf of the user, and continue to reuse the token credentials for as long as the user leaves his previous authorization in place.

First, here's what failure looks like. For example, when your request isn't signed with a user's token credentials.

```
In [38]: resource_url = 'http://api.mapmyfitness.com/3.1/users
/get_user'

In [39]: r = requests.get(url=resource_url)

In [40]: r.content
Out[40]: '{"result":{"status":-1,"output":[],"errors":["API Au
thentication failed for this Service. Bad API Key or Invalid U
ser."]}}\r\n'
```

For resource requests, the MapMyFitness API uses the Request URI Query method to receive OAuth parameters, as described in RFC 5849.

Sign a resource request with the user's token credentials to access resources.

```
In [41]: oauth = OAuth1(client_key, client_secret, unicode(tok
en_credentials['oauth_token'][0]), unicode(token_credentials['
oauth_token_secret'][0]), signature_type='QUERY')
```

```
In [42]: r = requests.get(url=resource_url, auth=oauth)

In [43]: r.content
Out[43]: '{"result":{"status":1,"output":{"user":{"user_id":"2
6345682","user_key":"26345682","user_type_id":"9","identifier"
:"","first_name":"A User","last_name":".","email":"a-user@exam
ple.com","username":"A User26345682","my_points":"0","my_point
s_award_level":"0","my_points_last_award_date":null,"profile":
"","photo_filename":"","thumbnail_filename":"","sex":"M","age"
:null,"birthdate":"1970-01-01","height":null,"weight":null,"re
sting_hr":null,"stride_length":null,"measurement_unit":"englis
h","temperature_unit":"F","nutrition_unit":"kCal","currency_id
":"20","preferred_date_format":"MM\\/DD\\/YYYY","first_day_of_
week":"sun","military_time_flag":"0","timezone_utc_offset":"0.
00","map_type":"3.1","start_address":"","start_city":"","start
_state":"","start_zip":"","start_country":"us","start_latitude
":null,"start_longitude":null,"activation_flag":"0","activatio
n_code":"","activation_date":"2013-01-16 04:45:40","system_opt
_out_flag":"0","system_opt_out_date":null,"opt_out_flag":"0","
opt_out_date":null,"promo_opt_out_flag":"0","promo_opt_out_dat
e":null,"interest_list":"","do_not_share_flag":"0","do_not_sha
re_date":null,"share_workout_flag":"0","use_advanced_workout_f
lag":"0","site_id":"11","site_skin_id":null,"site":"","promoti
on_code":null,"http_referrer":"","how_found":"web","visited_ne
w_site_flag":"0","last_login_date":"01\\/16\\/2013","last_logi
n_ip":"173.76.27.159","last_login_site_id":"11","last_login_ht
tp_referrer":"https:\\/\\/www.mapmyfitness.com\\/auth\\/reset_
password\\/foogi\\/3e6-598e0af8f7c8829cec66\\/","updated_date"
:"2013-01-16 07:56:42","created_date":null,"RoleID":"3","Style
ID":"1","CustomStyle":null,"VerificationKey":"","EmailVerifica
tionKey":null,"UtilizeEmail":"0","ShowName":"1","Icon":null,"P
icture":null,"Attributes":null,"CountVisit":"0","CountDiscussi
ons":"0","CountComments":"0","LastDiscussionPost":null,"Discus
sionSpamCheck":"0","LastCommentPost":null,"CommentSpamCheck":"
0","UserBlocksCategories":"0","DefaultFormatType":null,"Discov
ery":null,"Preferences":null,"SendNewApplicantNotifications":"
0","SubscribeOwn":"0","email_hash":"","Notified":"0","xref":nu
ll,"date_migrated":null,"er_user_id":null,"er_admin_user_id":n
ull,"available_credits":"0","affiliate_id":"0","membership_id"
```

:null,"privacy_setting":"1","privacy_password":"","training_privacy_setting":"0","training_privacy_password":"","status_id":"1","md5_password":"607c6606b3c26f719a204fdfc4ef3b7c","user_promotion_code":"","member_since_date":"01\\/16\\/2013","currency_code":"USD","currency_name":"U.S. dollar","currency_symbol":"$","route_engine_preference_json":"","notify_comment_added_flag":"1","allow_messaging_flag":"1","forward_messaging_flag":"0","display_activity_feed_flag":"1","display_professional_attributes_flag":"1","display_physical_attributes_flag":"1","display_personal_attributes_flag":"1","display_lifestyle_attributes_flag":"1","display_usage_stats_flag":"0","display_public_routes_flag":"0","user_match_type_list":"","nationality":"us","hometown":"","introduction":"","strengths":"","weaknesses":"","hobbies":"","highlights":"","gear_list":"","website":"","video_website":"","photo_website":"","music_website":"","body_type":"","waist_line":null,"shirt_size":"","appearance":"","best_feature":"","hair_color":"","eye_color":"","ethnicity":"","education":"","occupation":"","employment_status":"","income":"","marital_status":"","have_kids":"","want_kids":"","religion":"","attend_services":"","diet":"","smoking":"","drinking":"","activity_frequency":"","team_name":"","team_website":"","cycling_category":"","usa_cycling_license_number":"","running_category":"","send_to_twitter_flag":"0","twitter_username":"","twitter_password":"","send_to_facebook_flag":"1","facebook_username":"","facebook_password":"","sync_with_nike_flag":"0","nike_username":"","nike_password":"","tracker_security":"automatic","shipping_address":"","shipping_city":"","shipping_state":"","shipping_country":null,"shipping_zip":"","timezone":"America\\/New_York","id":null,"membership_type_id":null,"membership_level_code":null,"membership_cost":null,"next_billing_date":null,"expiration_date":null,"cancellation_reason":null,"updated_at":null,"site_name":"MapMyFitness","site_description":"MapMyFitness.com is a community web site for people who want to stay healthy, lose weight or train more effectively. MapMyFitness.com provides easy-to-use, comprehensive web-based mapping and logging tools to measure distance, count calories, and visualize your fitness activities.","action":"route","person":"member","site_link":"http:\\/\\/www.mapmyfitness.com","subdomain":"www","domain":"mapmyfitness.com","css_file":"www-mapmyfitness.css","header_file":"header.inc.php","footer_file":"footer.inc.php","

```
site_logo":"logo_mapmyfitness.gif","site_email":"MapMyFitness
<no-reply@mapmyfitness.com>","tracking_code":"UA-273418-51","d
efault_workout_type_id":"5","branding_id":"6","app_name":"iMap
MyFITNESS","email_header_logo":"http:\\/\\/static.mapmyfitness
.com\\/d\\/emails\\/notifications\\/assets\\/emailhead_logo_fi
tness.png","email_main_color":"#00338E","email_short_text":"FI
T","twitter_link":"http:\\/\\/mmf.cc\\/lr0Iwp","facebook_link"
:"http:\\/\\/mmf.cc\\/mxRuPn","resource_key":null,"start_locat
ion":"United States","profile_link":"\\/user_profile?u=2634568
2","training_link":"\\/user_training?u=26345682","unread_messa
ge_count":"0"}},"errors":[]}}\r\n'
```

Step 9: Client application: Do something just a little harder

Now that your client application can make requests on behalf of a user, you can do all kinds of things with the API.

Let's look at the user's workouts.

```
In [44]: r = requests.get(url='http://api.mapmyfitness.com/3.1
/workouts/get_workouts', auth=oauth)

In [45]: r.content
Out[45]: '{"result":{"status":1,"output":{"count":"0","workout
s":[]},"errors":[]}}'
```

There aren't any yet. Let's create one!

```
In [46]: r = requests.post(url='http://api.mapmyfitness.com/3.
1/workouts/create_workout', data={'user_id': 26345682, 'workou
t_description': 'a workout description', 'activity_type_id': 1
6, 'workout_date': '2013-01-16'}, auth=oauth)

In [47]: r.content
Out[47]: '{"result":{"status":1,"output":{"workout_id":2223990
46,"workout_key":"401135834983568979"},"errors":[]}}'
```

And have a look at the request headers and body that the client sent.

```
In [48]: r.request.headers
Out[48]:
{'Accept': '*/*',
 'Accept-Encoding': 'gzip, deflate, compress',
 'Content-Length': u'102',
 'Content-Type': 'application/x-www-form-urlencoded',
 'User-Agent': 'python-requests/1.1.0 CPython/2.7.3 Darwin/11.
4.2'}

In [49]: r.request.body
Out[49]: 'workout_date=2013-01-16&activity_type_id=16&user_id=
26345682&workout_description=a+workout+description'
```

Now if we look at the user's workouts again, we see the one we just created.

```
In [50]: r = requests.get(url='http://api.mapmyfitness.com/3.1
/workouts/get_workouts?user_id=26345682', auth=oauth)

In [51]: r.content
Out[51]: '{"result":{"status":1,"output":{"count":"1","workout
s":[{"workout_id":"222399046","workout_key":"40113583498356897
9","workout_date":"2013-01-16","workout_date_formatted":"Jan 1
6, 2013","completed_flag":"1","workout_privacy_setting":"0","w
orkout_type_id":"47","recurring_workout_id":null,"workout_type
_name":"Regular Run","parent_workout_type_id":"1","mets":"0","
workout_start_time":null,"workout_end_time":null,"workout_desc
ription":"a workout description","calories_burned":"0","route_
id":null,"route_key":null,"route_name":null,"distance":"0","nu
mber_of_repetitions":"1","time_taken":"0","weight":"0","effort
_level":"0","quality_level":"0","min_hr":"0","avg_hr":"0","max
_hr":"0","min_pace":"0","avg_pace":"0","max_pace":"0","min_spe
ed":"0","avg_speed":"0","max_speed":"0","min_power":"0","avg_p
ower":"0","max_power":"0","avg_cadence":"0","notes":"","number
_of_steps":"0","user_id":"26345682","user_key":"26345682","act
ivity_type_id":"16","updated_date":"2013-01-16 09:23:55","crea
ted_date":"2013-01-16 09:23:55"}]},"errors":[]}}'
```

What's Next?

Congratulations! You're ready to start using OAuth to access the MapMyFitness API in your own client applications. As you're developing, you can return to this tutorial and use the interactive client application in IPython to explore API requests and responses.

If you save the `token_credentials`, you can even reuse them to make resource requests without going through the web redirection flow again.

The tutorial client application can be a particularly useful tool as a point of comparison if you have any trouble getting OAuth 1 signing to work in your own real client application.

Check out the API documentation, and try a few more requests on your own.

If you still have questions, feel free to send them to api@mapmyfitness.com.