Package Index > **python-firebase** > **1.2**

python-firebase 1.2

Python interface to the Firebase's REST API.

Download
python-firebase-1.2.tar.gz

Python interface to the Firebase's REST API

build unknown

Installation

python-firebase highly makes use of the **requests** library so before you begin, you need to have that package installed.

```
$ sudo pip install requests==1.1.0
$ sudo pip install python-firebase
```

Getting Started

You can fetch any of your data in JSON format by appending '.json' to the end of the URL in which your data resides and, then send an HTTPS request through your browser. Like all other REST specific APIs, Firebase offers a client to update(PATCH, PUT), create(POST), or remove(DELETE) his stored data along with just to fetch it.

The library provides all the correspoding methods for those actions in both synchoronous and asynchronous manner. You can just start an asynchronous GET request with your callback function, and the method

To fetch all the users in your storage simply do the following:

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', None)
result = firebase.get('/users', None)
print result
{'1': 'John Doe', '2': 'Jane Doe'}
```

The second argument of **get** method is the name of the snapshot. Thus, if you leave it NULL, you get the data in the URL **/users.json**. Besides, if you set it to **1**, you get the data in the url **/users/1.json**. In other words, you get the user whose ID equals to 1.

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', None)
result = firebase.get('/users', '1')
print result
{'1': 'John Doe'}
```

You can also provide extra query parameters that will be appended to the url or extra key-value pairs sent in the HTTP header.

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', None)
result = firebase.get('/users/2', None, {'print': 'pretty'}, {'X_FANCY_HEADER': 'VERY FANCY'})
print result
{'2': 'Jane Doe'}
```

Creating new data requires a POST or PUT request. Assuming you don't append **print=silent** to the url, if you use POST the returning value becomes the name of the snapshot, if PUT you get the data you just sent. If print=silent is provided, you get just NULL because the backend never sends an output.

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', None)
new_user = 'Ozgur Vatansever'

result = firebase.post('/users', new_user, {'print': 'pretty'}, {'X_FANCY_HEADER': 'VERY FANCY'})
print result
{u'name': u'-Io26123nDHkfybDIGl7'}

result = firebase.post('/users', new_user, {'print': 'silent'}, {'X_FANCY_HEADER': 'VERY FANCY'})
print result == None
True
```

Deleting data is relatively easy compared to other actions. You just set the url and that's all. Backend sends no output as a result of a delete operation.

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', None)
firebase.delete('/users', '1')
# John Doe goes away.
```

Authentication

Authentication in Firebase is nothing but to simply creating a token that conforms to the JWT standarts and, putting it into the querystring with the name **auth**. The library creates that token for you so you never end up struggling with constructing a valid token on your own. If the data has been protected against write/read operations with some security rules, the backend sends an appropriate error message back to the client with the status code **403 Forbidden**.

```
from firebase import firebase
firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', authentication=None)
result = firebase.get('/users', None, {'print': 'pretty'})
print result
{'error': 'Permission denied.'}

authentication = firebase.Authentication('THIS_IS_MY_SECRET', 'ozgurvt@gmail.com', extra={'id': 123})
firebase.authentication = authentication
print authentication.extra
{'admin': False, 'debug': False, 'email': 'ozgurvt@gmail.com', 'id': 123, 'provider': 'password'}

user = authentication.get_user()
print user.firebase_auth_token
"eyJhbGciOiAiSFMyNTYiLCAidHlwIjogIkpXVCJ9.eyJhZG1pbiI6IGZhbHNlLCAiZGVidWciOiBmYWxzZSwgIml
hdCI6IDEzNjE5NTAxNzQsICJkIjogeyJkZWJ1ZyI6IGZhbHNlLCAiYWRtaW4iOiBmYWxzZSwgInByb3ZpZGVyIjog
InBhc3N3b3JkIiwgImlkIjogNSwgImVtYWlsIjogIm96Z3VydnRAZ21haWwuY29tIn0sICJ2IjogMH0.lq4IRVfvE
GQklslOlS4uIBLSSJj88YNrloWXvisRgfQ"

result = firebase.get('/users', None, {'print': 'pretty'})
print result
{'1': 'John Doe', '2': 'Jane Doe'}
```

Concurrency

The interface heavily depends on the standart **multiprocessing** library when concurrency comes in. While creating an asynchronous call, an on-demand process pool is created and, the async method is executed by one of the idle process inside the pool. The pool remains alive until the main process dies. So every time you trigger an async call, you always use the same pool. When the method returns, the pool process ships the returning value back to the main process within the callback function provided.

```
 import json
 from firebase import firebase
 from firebase import jsonutil

firebase = firebase.FirebaseApplication('https://your_storage.firebaseio.com', authentication=None)

def log_user(response):
    with open('/tmp/users/%s.json' % response.keys()[0], 'w') as users_file:
        users_file.write(json.dumps(response, cls=jsonutil.JSONEncoder))

firebase.get_async('/users', None, {'print': 'pretty'}, callback=log_user)
```

TODO

- Async calls must deliver exceptions raised back to the main process.
- More regression/stress tests on asynchronous calls.
- Docs must be generated.

File	Type	Py Version	Uploaded on	Size
python-firebase-1.2.tar.gz (md5)	Source		2014-03-21	9KB

Author: Ozgur Vatansever
Home Page: http://ozgur.github.com/python-firebase/
Keywords: firebase python
License: MIT

Categories

- **Development Status :: 5 - Production/Stable**
- **Environment :: Console**
- **Intended Audience :: Developers**
- **License :: OSI Approved :: MIT License**
- **Natural Language :: English**
- **Operating System :: OS Independent**
- **Programming Language :: Python :: 2.6**
- **Programming Language :: Python :: 2.7**
- **Programming Language :: Python :: 3.2**

Package Index Owner: ozgur
DOAP record: python-firebase-1.2.xml