lark Documentation

Release 0.1

Alex Kessinger

December 20, 2013

Contents

Lark is first and foremost a RESTy interface for redis. At it's core it's just a way of transforming HTTP requests into redis commands, but it comes with a few additions to make this a little more sane. It comes with adapters for Flask, but it would be simple enought to create an adapter that is suited for any other python webframework.

To get started make sure that you have redis installed, and then install lark.

```
pip install lark
```

Next you can create the minimalist of all Flask apps

```
from flask import Flask
from lark.ext.flask.redis_api import redis_api_blueprint
from lark.ext.flask.flask_redis import Redis

app = Flask(__name__)
# Add a simpple redis connection to the global object
Redis(app)

app.config['DEFAULT_LARK_SCOPES'] = set(['admin'])

# Mount the redis blueprint
app.register_blueprint(redis_api_blueprint, url_prefix='/api/0')


if __name__ == '__main__':
    app.run()
```

From here you can run the server and then you will be able to interact with the API like so. You can find documentation on all the calls here.

```
>>> curl http://127.0.0.1:5000/api/0/get/a/
{"meta": {"status": "ok", "status_code": 200}


>>> curl -X POST -H 'Content-Type: application/json' \
--data-ascii '{"value": "foo"}' \
http://127.0.0.1:5000/api/0/set/a/
"meta": {"status": "ok", "status_code": 200}, "data": true}

>>> curl http://127.0.0.1:5000/api/0/get/a/
{"meta": {"status": "ok", "status_code": 200}, "data": "foo"}
```

all of this is pretty cool, but this wouldn't be very usefull with out access control. So, there is a lark app dedicated to manage oauth2 apps so you can genearate access tokens with specific scopes.