# moth Documentation

***Release 2.1.1***

**Charles Thomas**

February 08, 2014

Contents

Contents:

CHAPTER 1

# Moth

Here's how standard moth works.

**class** moth.**Moth**(*database='moth'*, *host='localhost'*, *port=27017*, *user=None*, *pwd=None*)

Moth requires the credentials to log in to MongoDB.

**auth_token**(*email*, *token*, *ip=None*)

Return True if email address and token match. If IP exists, also verify that. If expiration was set when create_token was called, verify that the token hasn't expired. If for any reason the token is not valid, remove it.

**create_token**(*email*, *ip=None*, *expire=None*, *token_size=64*, *retval=None*)

Generate a token of a given length, tied to email address, and store it. Optionally store IP address, expiration (in days), and retval (see set_retval for additional information on this). Return the token

**fetch_retval**(*email*)

If retval exists, return it. If it doesn't, return True.

**remove_token**(*email*, *token*)

Remove token from Moth.

**remove_user**(*email*)

Remove all user data from Moth.

**set_retval**(*email*, *retval*)

Store retval associated with the email address. When auth_token is called, if the authentication was successful, and a retval exists, it will be returned by the auth_token call. If retval does not exist, auth_token returns True.

CHAPTER 2

# AsyncMoth

Here's how AsyncMoth works

**class** moth.**AsyncMoth**(*database='moth'*, *host='localhost'*, *port=27017*, *user=None*, *pwd=None*)

AsyncMoth blocks on __init__ when opening the MongoDB connection. All params are for creating said connection.

AsyncMoth is functionally equivalent to Moth, but with asynchronous support for use with Tornado/Motor.

**auth_token**(**args*, ***kwargs*)

Return True if email address and token match. If IP exists, also verify that. If expiration was set when create_token was called, verify that the token hasn't expired. If for any reason the token is not valid, remove it.

**create_token**(**args*, ***kwargs*)

Generate a token of a given length, tied to email address, and store it. Optionally store ip address, expiration (in days), and retval (see set_retval for additional information on this). Return the token.

**fetch_retval**(**args*, ***kwargs*)

If retval exists, return it. If it doesn't, return True.

**remove_token**(**args*, ***kwargs*)

Remove token from Moth

**remove_user**(**args*, ***kwargs*)

Remove all user data from Moth

**set_retval**(**args*, ***kwargs*)

Store retval associated with the email address. When auth_token is called, if the authentication was successful, and a retval exists, it will be returned by the auth_token call. If retval does not exist, auth_token returns True.

# WARNING

**DO NOT USE THESE EXAMPLES AS IS**

They are **UNTESTED**.

CHAPTER 4

# Using Moth to Authenticate Users by Email

```
#!/usr/bin/env python
# THIS CODE HAS NOT BEEN TESTED AND MAY NOT WORK.
# EVEN IF IT DOES, YOU SHOULDN'T RUN IT.
# IT IS INTENDED TO BE A GUIDE, NOT AN IMPLEMENTATION.

from base64 import b64decode, b64encode
from smtplib import SMTP

from tornado.web import RequestHandler

from moth import Moth


class LoginHandler(RequestHandler):
    moth = Moth()

    def get(self):
        x = self.get_argument('x', '')
        if x == '':
            self.write('''<html><body><form method=POST>Enter your email: <input
                       type=email><input type=submit></form></body></html>''')
        else:
            email, token = b64decode(x).split('&')
            email = email.split('=')[1]
            token = token.split('=')[1]

            if self.moth.auth_token(email=email, token=token) == True:
                self.set_cookie('email', email)
                self.redirect('/dashboard')
            else:
                self.redirect('/login')

    def post(self):
        email = self.get_argument('email')
        fromaddr = "noreply@moth.com"

        token = self.moth.create_token(email=email, expire=1)
        auth_string = b64encode("user=%s&token=%s" % (email, token))

        login_url = "https://login.moth.com/auth?x=%s" % auth_string

        message = "From: %s\r\nTo: %s\r\n\r\nclick to log in:\n%s" % \
            (fromaddr, email, user['fname'], login_url)
```

```
44
45          mail_server = SMTP('localhost')
46          mail_server.sendmail(fromaddr, email, message)
47
48          self.write('You should be receiving a login link shortly')
```

CHAPTER 5

# Using Moth to Authenticate Sessions

```
#!/usr/bin/env python
# THIS CODE HAS NOT BEEN TESTED AND MAY NOT WORK.
# EVEN IF IT DOES, YOU SHOULDN'T RUN IT.
# IT IS INTENDED TO BE A GUIDE, NOT AN IMPLEMENTATION.

from tornado.web import RequestHandler

from moth import Moth

class BaseHandler(RequestHandler):
    def get_current_user(self):
        email = self.get_cookie('email', '')
        session_token = self.get_cookie('session', '')
        if email == '' or session == '':
            return False

        return Moth().auth_token(email=email, token=session_token)
```

CHAPTER 6

# Making Asynchronous Calls with AsyncMoth

```
#!/usr/bin/env python
import logging

from tornado import gen
from tornado.ioloop import IOLoop
from tornado.web import Application, RequestHandler

from moth import AsyncMoth

class ExampleHandler(RequestHandler):
    @gen.coroutine
    def get(self):
        email = 'ch@rlesthom.as'
        want_retval = 'test retval'
        moth = server.settings['moth']

        ### create_token ###
        self.write('test create_token...<br>')
        token = yield moth.create_token(email, retval=want_retval)
        self.write(token + '<br>')

        ### auth_token ###
        self.write('test auth_token...<br>')
        have_retval = yield moth.auth_token(email, token)
        self.write('"%s" should match "%s"<br>' % (want_retval, have_retval))

        ### auth_token for invalid email ###
        self.write('test bad auth...<br>')
        should_be_False = yield moth.auth_token('fake@f.com', token)
        self.write('"%s" should be False<br>' % should_be_False)

        ### remove_token ###
        self.write('test remove_token...<br>')
        yield moth.remove_token(email, token)

        ### remove_user ###
        self.write('test remove_user...<br>')
        yield moth.remove_user(email)

        self.finish()

server = Application([('/', ExampleHandler)], debug=True)
server.settings['moth'] = AsyncMoth('moth_test')
```

```
44 server.listen(9000)
45 IOLoop.instance().start()
```

CHAPTER 7

# moth

**Moth** was conceived to be used as an email-only authentication scheme (mail auth -> mauth -> moth), however, it is generic enough to be used for pretty much any token-based authentication.

## 7.1 Relevant Links

- moth in PyPI
- moth on Github
- moth documentation
- moth build status
- moth author

## 7.2 Synchronous vs. Asynchronous

To create a synchronous moth object:

```
from moth import Moth
moth_object = Moth()
```

To create an asynchronous moth object:

```
from moth import AsyncMoth
moth_object = AsyncMoth()
```

All method calls are supported and identically named in **Moth** vs. **AsyncMoth**. For the rest of this README, **Moth** will refer to both **Moth** and **AsyncMoth** unless stated otherwise.

## 7.3 Initialization

Initiating **Moth** takes the credentials for creating a connection to MongoDB, as well as the database name (which defaults to "moth").

`AsyncMoth.__init__()` **blocks while creating a connection.** It is the only method which does so. It is recommended that you initialize **AsyncMoth** as part of your tornado server's startup.

## 7.4 Creating Tokens

Calling `moth.create_token()` generates a random token and stores it along with email address and optional IP address, expiration (in days), and retval. The method returns the token.

## 7.5 Authenticating Tokens

Calling `moth.auth_token()` queries mongo for the passed email/token combination. If IP address is in the record returned from mongo, it is validated. If expiration is returned, it is compared to `datetime.now()`

**If either IP address or expiration fails to validate, the token will be deleted.**

If the token validates, retval is queried. If a retval exists, it is returned. If it doesn't, `moth.auth_token()` returns True.

## 7.6 Additional Methods

All other methods are fairly self explanatory, and/or mostly for internal purposes. Read the code to figure out how it works.

## 7.7 What is retval?

retval is the value that will be returned when `moth.auth_token()` is successful. It is **completely optional**. If you don't pass a retval to `moth.create_token()`, and don't call `moth.set_retval()`, then `moth.auth_token()` will return True on successful calls.

### 7.7.1 Why use it?

For the project I'm working on which lead to the creation of **Moth**, retval is an OAuth token. When I call `moth.auth_token()`, I validate the moth token, which gives me the user's OAuth token for making API calls.

## 7.8 Requirements

**Moth** requires **Motor**, as well as **Tornado** and **Pymongo** (which are both installed via **Motor**).

## 7.9 Examples

Examples can be found on ReadTheDocs

CHAPTER 8

# Indices and tables

- *genindex*
- *search*

## m