ADNPy Documentation

Release 0.2.1

Alex Kessinger

December 19, 2016

Contents

ADNPy aims to be a easy to use library for interacting with the App.net API.

CHAPTER 1

Installation

```
pip install adnpy
```

You may also use Git to clone the repository from Github and install it manually:

```
git clone https://github.com/appdotnet/adnpy.git
python setup.py install
```

CHAPTER 2

Quick Start

ADNPy aims to be an easy-to-use Python client for the App.net API. To get started, you'll need an access token, which you can get by creating an app on App.net and clicking the "Generate a user token" link.

```
import adnpy

# Set the default access token for API calls.
adnpy.api.add_authorization_token('your access token here')

# Send a broadcast with the BroadcastMessageBuilder recipe.
builder = adnpy.recipes.BroadcastMessageBuilder(adnpy.api)
builder.channel_id = 24204  # Get this channel ID from the web publisher tools
builder.headline = 'Hello World!'
builder.text = 'Sending this from [ADNPy](https://github.com/appdotnet/ADNPy) was easy!'
builder.parse_markdown_links = True
builder.read_more_link = 'http://adnpy.readthedocs.org/'
builder.send()

# Or create a post using the API module.
post, meta = adnpy.api.create_post(data={'text': 'Hello World'})
```

CHAPTER 3

Reference

3.1 Getting started

3.1.1 Introduction

ADNPy aims to be an easy-to-use Python client for the App.net API. To get started, you'll need an access token, which you can get by creating an app on App.net and clicking the "Generate a user token" link.

3.1.2 The Obligatory Hello World Example

```
import adnpy

# Add an access token to authorize access to your account
adnpy.api.add_authorization_token('your access token here')

# Create a post
post, meta = adnpy.api.create_post(data={'text': 'Hello World'})
```

This example creates a simple post on your App.net Stream.

3.1.3 Recipes

Recipes are simple ways to use the App.net API which shield you from having to learn about resources, entities, annotations, and all of that – find out more on the *Recipes* page.

3.1.4 API

The API class provides methods for each App.net endpoint. To find more information about each method, check out the *API Reference*.

3.1.5 Models

API calls normally return APIModel objects. There's an APIModel for each common App.net resource, like Post, User, and Channel. APIModels are essentially just subclasses of *dict*, with extra syntacical sugar allowing you to access information via dot notation or normal dict methods:

```
# given a post object
print post.text
print post.get('text')
print post.get('user', None)
```

Each model can also preform certain API calls customized to the current model instance:

```
post.delete()
```

For more information about models, please see the *Model Reference*.

3.2 Recipes

Simple recipes for using the App.net API.

3.2.1 `adnpy.recipes` — Simple App.net API Recipes

3.3 Model Reference

This page contains some basic documentation for the ADNPy Models

3.3.1 `adnpy.models` — App.net API Models

3.4 API Reference

This page contains some basic documentation for the ADNPy API

3.4.1 `adnpy.api` — App.net API

3.4.2 User Methods

A User is the central object of the App.net APIs. User objects have usernames, follow other users, and post content for their followers. See User Developer Docs.

3.4.3 Post Methods

A Post is the other central object utilized by the App.net Stream API. See Post Developer Docs.

3.4.4 Channel Methods

A Channel is a user created stream of Messages. It controls access to the messages in the channel allowing for (among other things) public, private, and group messaging. See Channel Developer Docs.

3.4.5 Message Methods

A Message is very similar to a Post but 1) it doesn't have to be public and 2) it will be delivered to an arbitrary set of users (not just the users who follow the Message creator). See Message Developer Docs.

3.4.6 File Methods

A file is uploaded by a User and hosted by App.net. See File Developer Docs.

3.4.7 Interaction Methods

Interactions are objects that represent users taking certain actions on App.net. See Interaction Developer Docs.

3.4.8 Text Process Methods

When a request is made to create a Post or Message, or update a User profile description, the provided body text is processed for entities. You can use this endpoint to test how App.net will parse text for entities as well as render text as html. See Text Process Developer Docs.

3.4.9 Token Methods

Returns info about the current OAuth access token. If the token is a user token the response will include a User object. See Token Developer Docs.

3.4.10 Config Methods

3.4.11 Place Methods

Place objects represent physical locations which can be given a name and associated with a latitude and longitude somewhere on Earth. See Place Developer Docs.

3.4.12 Explore Streams

An Explore Stream is a subset of all public posts flowing through App.net's Global Stream. These Explore Streams are defined by App.net to provide developers and users new ways to discover posts. See Explore Streams Developer Docs.

3.4.13 Config Method

The Configuration object contains variables which define the current behavior of the App.net platform. See Config Developer Docs.

3.4.14 App Stream Methods

A customized view of the global events happening on App.net that is streamed to the client instead of polling. See App Stream Developer Docs.

3.4.15 Filter Methods

A Filter restricts a stream of messages on the server side so your client only sees what it's interested in. Streams are currently the only way to use filters right now. See Filter Developer Docs.

3.5 Streaming Reference

This page contains some basic documentation for the ADNPy streaming interface. Check out the docs for more on how to use app streaming.

3.5.1 `adnpy.stream` — App.net App Streaming

a

A