"Python access to TD Ameritrade's options data using application programming interfaces"

Daniel H. Miller
dhmmhd at at at yahoo.com
Presented to AAII Options, San Diego, CA, Aug 28, 2021, 09:00 to 11:00

Abstract:

Most option traders look at option chains on the TD Ameritrade's ThinkOrSwim platform. This talk presents background and specific examples of how to use the Python programming language to access option chains using application programming interfaces (APIs) accessed via the REST-style (REpresentational State Transfer) protocol. The presentation will provide an overview of REST-style API's in general, specific APIs provided by TD Ameritrade, and finally a few Python code examples that use options data for specific illustrative trading methods. Access to this programmatic data is available for free to TD Ameritrade customers. Likewise, Python and the programming environment (Anaconda) are open source and freely available for multiple operating systems.

Who I've Worked With on This Project

* Ryan McKeon, former leader of AAII Options San Diego, former finance professor at University of San Diego

* Corbin Miller, fellow option trader, and Apple-oriented/iPhone Swift programmer, who used a different programming language but dealt with the same problems of authentication

Speaker Background

* Background in math/computer science, so oriented to quantitative approach to options trading

* However, I'm not a hard-core programmer

* Long-time open source/Linux user, and so is all the software used in this project

* Silicon Valley colleague introduced me to options on ThinkOrSwim in 2006

Agenda

* Give some background about programmatic access to the communications protocols needed to retrieve options data at TD Ameritrade

* Explain some specific Python language set up and TDA configuration

needed to access options data at TD Ameritrade

* Show a few examples of Python code that access options data at TD Ameritrade (cover as many as time permits)

* Take questions along the way, and at the end

* Open discussion about applications of the options data

Emphasis

* More like a show and tell, rather than a one-way talk

* Talk 25% of the time vs. demonstrate Python software 75% of the time

* Discussion 50% vs. demo 50%

Broker/Platform Scope

* This presentation revolves around TD Ameritrade's (TDA) ThinkOrSwim (ToS) and the underlying data obtained from TDA

* Other brokers may have their own programmer oriented data services, so many of the concepts of this talk will apply,
  but likely not the details

Why Use a Program Rather than the ToS Platform

* Data quantity: Compare all $500 risk verticals along an options chain, for all expiration cycles between now and 70
  days from now

* Speed: Look at prices in real-time to determine when pricing is favorable

REST High Level/Overview

* RESTful web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the
  use of JSON or XML to transmit data. (Wikipedia: "Representational state transfer")

* [REST describes] how a well-designed Web application behaves: it is a network of Web resources (a virtual state
  machine) where the user advances through the application by selecting links (e.g. http://www.example.com/articles/21),
  resulting in the next resource's representation (the next application state) being transferred to the client and
  rendered for the user. (Wikipedia: "Representational state transfer")

* Goal of REST is to increase performance, scalability, simplicity, modifiability, visibility, portability, and
  reliability. This is achieved through following REST principles such as a client–server architecture, statelessness,
  cacheability, use of a layered system, support for code on demand, and using a uniform interface. These principles
  must be followed for the system to be classified as REST. (Wikipedia: "Representational state transfer")

REST and APIs

* APIs are "Application Program Interfaces", which are collection of actions that can be requested by one program/system
  to another program/system, usually across a (Internet) network

* How to use REST (REpresentational State Transfer) is a style of API/protocol commonly used on the Internet

** Your local machine sends queries over the (Internet) network to a server containing (large) database against which
   you create a (relatively small) query/subset

** REST queries are transmitted using a stylized http: request (yes, just like your browser does)

** REST endpoints are queries that are run on the server, endpoints have unique names/arguments

** REST endpoints output is provided by the server after running the endpoint code and sends a reply from the server
   back the client, reply is usually in JSON (JavaScript Object Notation) format

** REST responses in JSON are parsed/decomposed by the client/local program

Examples and Range of Use of REST

* Simple weather example: Program on your computer asks for current weather conditions in London from a weather server,
  whose response includes local time, local temperature, wind direction, etc, of which you extract and display (only)
  city name, date/time, and temperature

* Simple Twitter example: https://api.twitter.com/2/tweets/search/recent?query=from:bob gets recent tweets posted by
  user Bob

* REST is a protocol, so any of numerous languages can be used to

generate REST queries and then parse/use the JSON
  responses

* REST queries are constructed as formatted http: requests

** Languages like Python have helpful libraries for conveniently creating and sending REST queries and
   processing/parsing JSON responses (and whose details are thus not necessary for us to understand in order to write
   our software)

** REST endpoint queries can be created (by hand or otherwise) within a web browser, and and the JSON response viewed as
   a web page

* Calling an endpoint is like calling a function, but over the network: Calling an endpoint requires the endpoint's
  name, and required/optional parameters

TDA's REST API

* Done once: Anyone who has a TDA trading account uses a separate and free account at
  https://developer.tdameritrade.com/ to initialize access to TDA's options data (involves creating an account, an
  "app"lication, and obtaining initial OAuth 2.0 security tokens)

* Done each time a program wants to access options data: authentication

** Prior to being able to make REST queries against TDA's servers, the application must log in ("authenticate") using
   OAuth2.0 to gain access to TDA data

** This security functionality is provided seamlessly by an open source package called "tda-api"

** Note: I wanted to focus on options trading rather then learning the details of "OAuth 2.0" authentication protocol

* TDA sample end point of interest: Stock price, "what is the current stock price of IBM?" (note: current and history
  endpoints)

* TDA sample end point of interest: Option chains, all the attributes seen as columns on the ToS platform, for example:
  "what is the option chain for IBM expiring on Sep 18, 2021?" (note: only current data, no history available)

* Important: Historical stock quotes can be obtained, but not for

option chains (this would be VERY nice to have)

* These are NOT statistics about options (number of put/call options transacted today)

* During market hours, data (quotes, option chains) are real-time

* Real-time limits: "All non-order based requests by personal use non-commercial applications are throttled to 120 per
  minute. Exceeding this throttle limit will provide a response with a 429 error code to inform you that the throttle
  limit has been exceeded."

* After market hours, data are end of day

* Each TDA endpoint is documented on a TDA web page which also assists in creating a valid REST query, and then displays
  the returned JSON, which is very convenient for becoming familiar with endpoints, debugging authentication, and
  examining returned JSON structures

Illustrating TDA Endpoints on developer.tdameritrade.com

* Create a log-in at https://developer.tdameritrade.com/

* Create an "app": Each app has a long alphanumeric key associated with it (part of OAuth2.0)

* Categories of endpoints (https://developer.tdameritrade.com/apis)

** Accounts and Trading: APIs to access Account Balances, Positions, Trade Info and place Trades
** Authentication: oAuth API to retrieve the bearer token which can be used to access other APIs
** Instruments: Search for instrument and fundamental data
** Market Hours: Operating hours of markets
** Movers: Retrieve mover information by index symbol, direction type and change
** Option Chains: Get Option Chains for optionable symbols
** Price History: Historical price data for charts
** Quotes: Request real-time and delayed top level quote data
** Transaction History: APIs to access transaction history on the account
** User Info and Preferences: APIs to access user-authorized accounts and their preferences
** Watchlist: APIs to perform CRUD operations on Account Watchlist

Benefits of using TDA's Website Interface to Construct REST Queries

* TDA provides a web based service to create REST queries for the user

* This helps the user understand the range of required vs. optional parameters

* Useful for initial review of content and format of (JSON) response

Mechanics of Using TDA's Website Interface to Construct REST Queries

* Each endpoint has its TDA web page (samples to be reviewed below)

* Fill in arguments (beware: sometimes it is difficult to know which parameters conflict with other parameters, example:
  different ways to specify an expiration to obtain an option chain)

* Examine "Response Summary" documentation

* Submit

* Review "Request" tab: "GET" request, formatted as http: request using ? notation for each argument

* Examine "Response" tab

* Confirm "success" response

* Review JSON response content (e,g., stock quote, option chain and it horrible format)

Sample Endpoints

* Sample endpoint: Get a current stock quote

** REST style URL: https://api.tdameritrade.com/v1/marketdata/{symbol}/quotes

** TDA Ameritrade web page that constructs a REST URL in the browser:
   https://developer.tdameritrade.com/quotes/apis/get/marketdata/{symbol}/quotes

* Sample endpoint: Get current option chain

** REST style URL: https://api.tdameritrade.com/v1/marketdata/chains

** TDA Ameritrade web page that constructs a REST URL in the browser: https://developer.tdameritrade.com/option-chains/apis/get/marketdata/chains

API Key: (Select app from pulldown)
Symbol: IBM
contactType: ALL

strikeCount: 10
includequotes: FALSE
strategy: SINGLE
fromDate: 2021-09-04
toDate: 2021-09-28
optionType: S

Anaconda: Python Development Environment

* Python language interpreter is open source and free

* Anaconda is a self-contained (and expandable) distribution of Python that includes the language interpreter as well as
  numerous key/core libraries to do common tasks

* Ananconda also provides a simple Interactive Development Environment (IDE) including a language sensitive editor
  called Spyder to make programming more interactive

* Programming projects are stored in "environments" which have their own/versions of software

Using Anaconda for TDA Projects

* Keep Anaconda software up to date: conda update -n base -c defaults conda

* Create project (sample name "tda"): conda create --name tda --clone base

* Make project current environment: conda activate tda

* Keep software within project environment up to date: conda update --all

* Install "tda-api" module software: pip install tda-api

* Keep "tda-api" module up to date: pip install --upgrade tda-api

* Confirm software updates do not break current version of software (which it did recently, when they changed the names
  of some end point arguments)

* First: Confirm ability to authenticate

Overview/Style of Sample Programs

* "login" using OAuth2.0 authentication, using "tda-api" module

* Perform query (stock, option chain), using "tda-api" module

interface to REST

* Parse apart response into manipulable format (Python Pandas data frame, like a spreadsheet table)

* Use the stock/options data in a trading method

Sample Python Programs using TDA Data

* Data retrieval
** AAPL stock price history
** SPY option chain for a few specified month range, 2 strikes around the money
** SPY option chain for a single specified month August, 2 strikes around the money
** SPY option chain for a few months (all strike prices, all/weekly expiration cycles)

* Data manipluation
** For each expiration and side calls vs. puts, output cumulative open interest
** Combined weighted average open interest based predictor (CWOP)
** All possible iron condors

Using TDA's Data Yourself

* Setup using "Using Anaconda for TDA Projects"

* Copy sample code from "tda-api" and/or samples shown today

* Start programming yourself: Really, the only way to learn

** Start with small examples

** Build bigger/more ambitious programs

Documentation/Software

* TD Ameritrade Developer: https://developer.tdameritrade.com/
(includes critical general guides to security, usage, and
  documentation for each endpoint)

* "tda-api" Python module to handle OAuth2.0 authentication and access to data:
  https://tda-api.readthedocs.io/en/latest/index.html, https://github.com/alexgolec/tda-api/

* REST (REpresentational State Transfer): https://restfulapi.net/
(just an example/overview),
  https://en.wikipedia.org/wiki/Representational_state_transfer

* Anaconda development environment for Python (includes Python interpreter): https://www.anaconda.com/

Discussion/Opening Questions

* If you had real-time options chains, what trading methods would want to program that could not performed by hand or
  could not be perform fast enough manually?

* Since TDA does not provide option chain history, would it be worth it to create your own database from their data?
  (Warning: This may not be permitted under agreements, I don't know)