pyGeodashboard

Development of pyGeodashboard started on 2016-02-04. It is a library that contains the basic functions needed for parsing sensors, streams, and datapoints to the geostreaming API.

- Step 1: Outline Parser Functions
  - Create a outline that describes the process of parsing with focus on separating reusable portions of a parser from those that are particular to a specific data source.

Outline Parser Functions

Functions will be described as *unique*= code particular to the source and *general*= should be able to run as part of every source

1. Get data from source (unique)
    a. Parsing begins by getting the data from the source. Two types of data are needed:
        i. data that describes the site such as geocodes, name, and source.
        ii. measurements (the data)
    b. The format and retrieval method varies from source to source
        i. Some source formats
            1. API for a single station (USGS,NOAA,EPA)
            2. API for mixed stations (Water Quality Portal)
            3. Files stored to server with loggetnet (GREON)
            4. csv download (LRTM)
2. Parse data to sensor (unique and general)
    a. Up till now, this has been a unique process for each source; however, this portion should be broken into unique and general portions
        i. reformat data into a standard that can be input into general parser (unique)
        ii. parse data to sensor json (general)
        iii. post to geostreaming api (general)
3. Parse data to stream(s) (unique abd general)
    a. Similar to parse to sensor, with the main difference being that sources can have multiple stream for different reasons.
        i. For example:
            1. GREON uses 2 streams - one for water quality data and one for environmental data
            2. USGS uses 5 streams - water quality measurements, gap filled nitrate, gap filled discharge, load, and cumulative load
        ii. two different conventions have been used, and need to be standardized
            1. GREON names the streams differently: GREON-07_MD or GREON-07_WQ
            2. USGS puts a data_type key in properties with possible values: source_data, fill_nitrate, fill_discharge, calc_load, and calc_cumul_load
                a. Probably should be discussed and decided.
    b. Currently, each source has it's own implementation, like sensors, and should be broken into unique and general portions
        i. get stream data from source including determining number of streams needed and parsing data to standard format (unique)
        ii. parse data to stream json (general)
        iii. post to geostreaming api
4. Parse data to datapoints (unique and general)
    a. Initial parse versus continuous parsing
        i. Parsing data to a site continuously (parsing at a regular interval) has some challenges
            1. need to know the last datapoint parsed (general)
            2. more efficient to only fetch source data from that time forward (unique)
    b. Considerations
        i. Time - keeping track of time zone and daylight savings time is challenging
            1. The geostreaming api stores time in UTC (Zulu)
                a. sources store time in a variety of formats
            2. The suggested standard: all retrieved data should be converted to UTC as soon as possible within the scripts and handled in UTC.
        ii. Memory
            1. The earliest versions of the parsers loaded all data into memory from a single station and parsed from memory
                a. after adding new usgs sites, it became apparent that this would not work on a server with ~2GB memory (or even larger as datasets grow)
                b. 2 approaches were considered: disk storage and parsing by time period
                c. it was decided to parse one year at a time
    c. Parsing
        i. fetch data from source (unique)
        ii. parse to standard format (unique)
            1. map measurement names to standardized parameter ids
        iii. parse to datapoint json (general)
        iv. post to geostreaming api (general)
5. Gap filling and load calculations
    a. At the moment, these are only run on USGS data
    b. This should be generalizable to any stream that contains the needed parameters
        i. Gap Filling: should take the source data stream and fill parameter
        ii. Load calculation: should take the source data stream and gap fill streams as input
        iii. Cumulative load calculation: should take the load stream as input
6. List of methods that should be general (or have several general versions)
    a. create or get sensor/stream
    b. parse to sensor json
    c. parse to stream json
    d. parse to datapoint json
    e. post
    f. get

g. get most recent datapoint
h. iterate over a year of data
i. convert time
j. map_names
    i. needs to be standardized across sources with different column names mapped to standard parameter ids