Funhandler Developer Docs

Funhandler is the datahander for the live testing library. The overall idea behind this library is that the user will be able to enter new information for the backtest of the system.

It's meant to be the the modual version of the data handler inside of the backtesting tutorial:

https://www.quantstart.com/articles/Event-Driven-Backtesting-with-Python-Part-III

The general idea works like the following:

1. The user is able to set the location of the database (a mongodb database)
2. The user sends a dataframe representing the data that they need to save.
3. The data is saved into the mongo database.
4. To backtest we load the data of a trade pair based on a given timeframe.
5. We can begin pulling the latest time for a given trade pair.
6. If we have no more bars loaded in the latest pair return false to indicate the cycle is finished.

Desired Use Case

```
# Import Funhander
import funhandler as fh

MONGOHOST = 'localhost'

fh.set_host(MONGOHOST)

# This is the data for a given time period
df = pandas_data

# Adds the trading bars into the system. (funtime)
fh.add_bars(df)


# Loads a series of bars into the file
fh.load_bars('BTC', 'USD', 'binance')

# This will return false if there are no more bars available
while fh.is_still_bars('BTC', 'USD', 'binance'):
    # Should have a dataframe to get the latest bar for the user to analyze
    barframe = fh.get_latest_bar('BTC', 'USD', 'binance') # Should be increasing as ths is_
```

Desired Inner Workings

Internally the inner workings should allow for scale. The reasoning is that we'd be able to run multiple backtest at the exact same time asyncronously. This will be so we can ensure our AI is well trained at scale. This means the following:

1. The added bars will have to exist inside of a database
2. The loaded bars will have to be stored somewhere for later access. We recommend parquet stored inside of a global store such as S3. The location of the saved file will be referrable inside of the funtime library.
    - This is so we can run through the effort of pulling the data at scale and with different sources at the same time.
    - With this design we could pull anything.
3. We would want to placed the latest bar inside of a parquet file as well. It should append the latest bar into the composite column data and return the relavant information for the backtest.

What You're Getting

The original set of code you have will have some parts of the overall working module. It'll have the necessary setters and getters so you(The developer), can create the necessary functions very quickly. The store will be working and the means of collecting the information will be provided by the library `funpicker`.

How to Run

This requires `pipenv` to run. From there the funhandler should work.

```
pipenv install -e .
```