**Welcome**

Ultra-finance is a pure Python library & utility for real time stock data collection, analyzing and backtesting.

**Code has been moved to** <https://github.com/panpanpandas/ultrafinance>

**First Build**

* First build is available at: <http://code.google.com/p/ultra-finance/downloads/list>
* Installation instruction can be found at: <http://code.google.com/p/ultra-finance/wiki/BuildProcess>
* Any questions can be posted at: <http://groups.google.com/group/ultra-finance?pli=1>

**Examples**

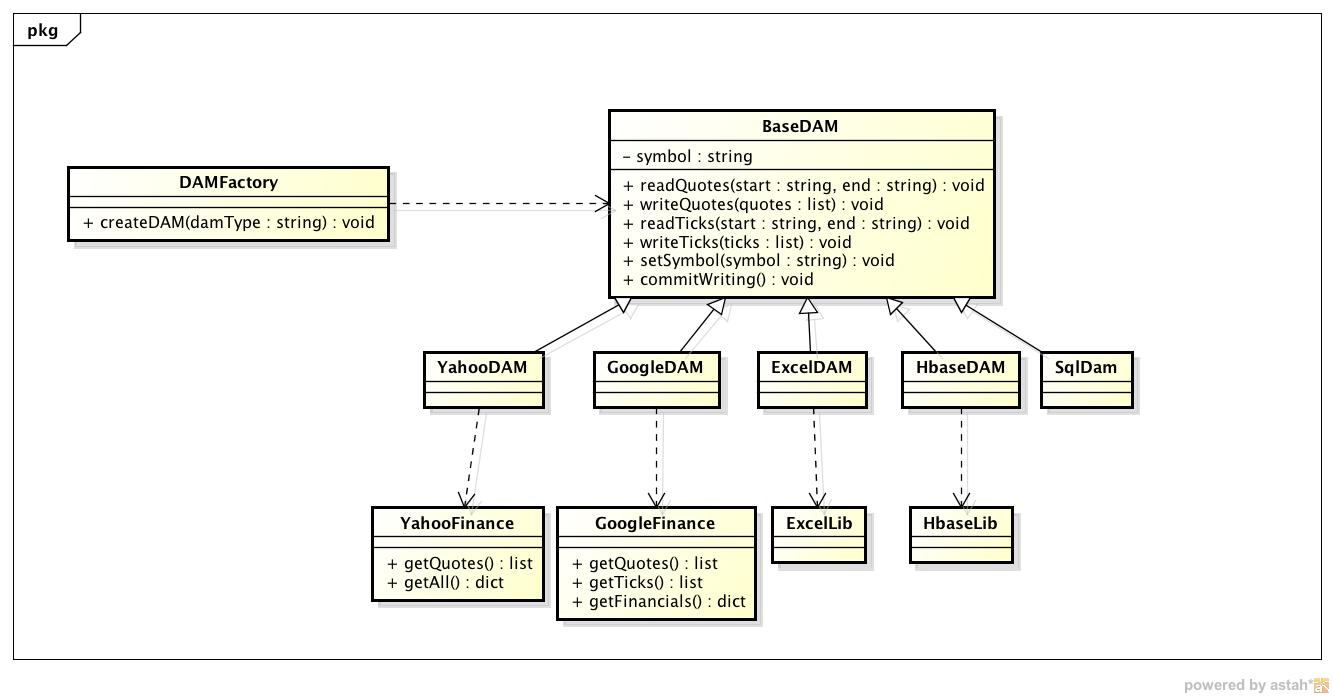
* [stock crawler](http://code.google.com/p/ultra-finance/wiki/stockCrawler) -- save stock quotes/ticks to local disk(sqlite or hbase)

**Design**

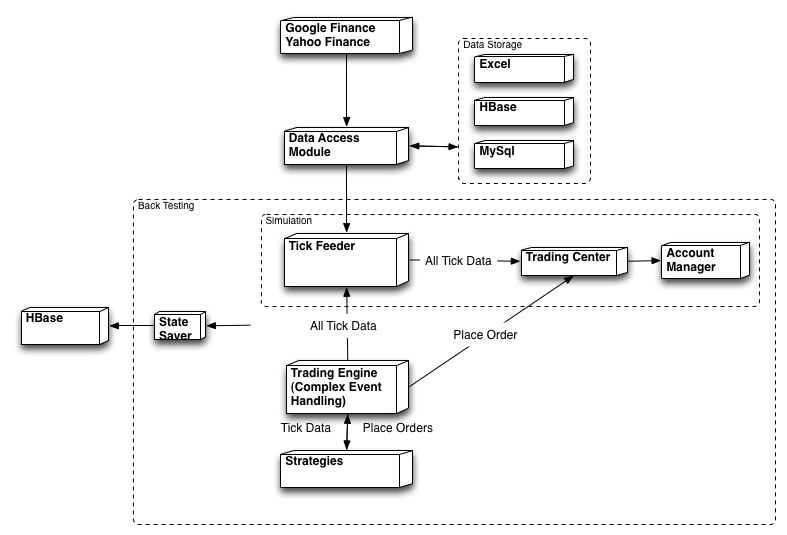
We try to keep the structure flat. Most modules are independent and can be executed easily.

Ultra-finance consists of four components:

* DAM(data access model): provides general API for quote/tick/financials access. There are four DAMs implemented: Yahoo Finance, Google Finance, HBase and Excel.



* Backtesting: validates trading strategies with historical quotes/ticks. Backtesting is functional completed now.



* Python TA-Lib: pure python library that process financial data. This component is still in design.
* Stock picker: Based on real data and user defined filter, this component will rank/pick the stocks.

**Interesting Result**

By using ultra-finance, we find something interesting: <http://code.google.com/p/ultra-finance/wiki/interestingResults>

**Features**

|  |  |  |
| --- | --- | --- |
| COMPONENT | FEATURE | STATUS |
| DAM |  |  |
|  | Retrieve quote/tick/financials from Yahoo Finance | DONE |
|  | Retrieve quote/tick/financials from Google Finance | DONE |
|  | Retrieve/Save quote/tick from Excel | DONE |
|  | Retrieve/Save quote/tick from HBase | DONE |
|  | Retrieve/Save quote/tick from MySql | DONE |
| Back Test |  |  |
|  | Tick Feeder Simulator | DONE |
|  | Trading Center Simulator | DONE |
|  | Account Manager | DONE |
|  | Trading Engine | DONE |
|  | Metrics -- Sharpe Ratio, Lowest, Highest | DONE |
|  | Metrics -- Average, Standard Deviation, Beta, Alpha | SCHEDULED |
|  | Strategy -- Period | DONE |
|  | Strategy -- MACD | DONE |
|  | Save Orders/Ticks/Position to HBase | DONE |
|  | Save Orders/Ticks/Position to Sqlite | DONE |
|  | Integrate with Sqlalchemy | DONE |
|  | Complex Event Processing | SCHEDULED |
|  | Generate Graphic Report | SCHEDULED |
|  | UNIT TEST | SCHEDULED |
| Design Patterns |  |  |
|  | Singleton | DONE |
|  | Observer | DONE |
|  | More patterns |  |
| Stock picker |  |  |
|  | Sort Stocks by Revenue Increasing | DONE |
|  | Sort Stocks by Diluted Normalized EPS | DONE |
|  | Save Financials to HBase | SCHEDULED |
|  | Build Stock Picker Framework Loads filter Plugins | SCHEDULED |
|  | Filter Plugins -- Sort Filter by Dividend, P/E ... | SCHEDULED |
|  | Unit Test | SCHEDULED |
| Python TA-Lib |  |  |
|  | Design Interface |  |
|  | FUNCTIONS -- ... | SCHEDULED |
| DEPRECATED |  |  |
|  | Plot any data and save it to image file | DONE |
|  | Send email to user (automatic sending alert to one's email address when filter triggered) | DONE |
|  | Optimize two risky portfolios | DONE |
|  | Config file/dynamic config for the program | DONE |
|  | Processing chain(based on event-driven) - multiple components hook together to achieve a complex task | DONE |
|  | Analyze 524 oversea Chinese stocks' return, alpha for 1 day, 1 week, 3 months and 1 year | DONE |
|  | **Stock measurement, including average, return rate, alpha, beta** |  |
|  | Stock average and standard deviation | DONE |
|  | Stock alpha and beta | DONE |
|  | **Trading strategy simulation** |  |
|  | Automatic investment plan: buy $1000 at the end of each year for SPY500(since year 1900). | DONE |
|  | Automatic investment plan: buy $1000 at the end of each year plus an addition $1000 if index is the lowest during last 3 years. | DONE |
|  | Automatic investment plan: buy $6000 per half a year, whenever the price is the lowest during the last half year, or just buy it at the end of the period | DONE |
|  | Weekly trading: buy and sell in one week | DONE |
|  | **Others** |  |
|  | Unit-test, exception handling and logging | DONE |
|  | Make first build | DONE |
|  | Write installation instruction on Windows, Linux and Mac | DONE |

Plan: In near future, this project will be focus on backtesting. The first step is to do backtesting on one strategy using one stock quote data as input. The final goal to do backtesting within a single run to get results of multiple strategies with hundreds of separate rounds while each round takes multiple stocks quotes/ticks as input.

Features/effort are listed below for each run of backtesting:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Tests | Number of Stock Inputs Per Test | Number of Strategies Per Round | Data Type | Realtime | Effort | Status |
| 1 | 1 | 1 | Quote | No | 2 full days | DONE |
| 1 | multiple | 1 | Quote | No | 2 full days | DONE |
| multiple | multiple | 1 | Quote | No | 2 full days | DONE |
| multiple | multiple | multiple | Quote | No | Unknown |  |
| multiple | multiple | multiple | Quote/Tick | Yes | Unknown |  |

Wish-list:

* UI or website or graph generating
* real time web news and analyze(web crawler), when bad/good news comes, ultra-finance will do trading automatically
* integrate with NumPy(data processing), PyQT(or wxpython), Pyramid(Pylons) and pandas