Разработка робота на Python

- Алго рантайм
- Справочное руководство Python Algo
  - Перечисления
    - xroad.TimeInForce
    - xroad.OrderFlag
    - xroad.OrderEvent
    - xroad.AssetEvent
    - xroad.EnvEvent
    - xroad.TradingStatus
    - xroad.MdataFeedStatus
    - xroad.OrderState
    - xroad.OrderType
    - xroad.PropsEvent
    - xroad.PropType
  - Глобальные функции
    - xroad.set_env_event(callback:fn) → None
    - xroad.send_msg(node: str|int, data::dict)
    - xroad,md_hbt_start() → None
    - xroad.md_hbt_stop() → None
    - xroad.start() → None
    - xroad.stop() → None
    - xroad.exit() → None
    - xroad.reconfig() → None
  - Классы
    - xroad.Asset
      - Asset(name::String, event:fn(Asset, AssetEvent, dict)) → Asset
      - name → str
      - total_buy → Float
      - avg_buy → Float
      - total_sell → Float
      - avg_sell → Float
      - md_stat → MdStat
      - trading_status → TradingStatus
      - ref → dict
      - md_quote(event::fn(quote:MdQuote), snapshot:Boolean, source:str) → MdQuote|None
      - md_info(event::fn(info:MdInfo), snapshot:Boolean, source:str) → MdInfo|None
      - md_book(event::fn(book:MdBook), snapshot::Boolean, source:str) → MdBook|None
      - md_trade(event::fn(trade:MdTrade), snapshot::Boolean, source:str) → MdTrade|None
      - md_indicator(event::fn(trade:MdIndicator), snapshot::Boolean, source:str) → MdIndicator|None
      - md_feed_status(event::fn(FeedStatus), snapshot::Boolean) → MdFeedStatus|None
      - md_stop() → None
      - cancel() → True
      - buy(name:str, account:str, client_code:str, qty:float, price:float, tif:TimeInForce, portfolio:str, flags:int, event:fn(OrderEvent, dict), till_date:datetime, send:Boolean, till_date:datetime.date, send:bool, type:OrderType, comment:str, collect_trades:bool, display_qty:int, broker:str, sender:str, user_data:Object) → Order
      - sell(name:str, account:str, client_code:str, qty:float, price:float, tif:TimeInForce, portfolio:str, flags:int, event:fn(OrderEvent, dict), till_date:datetime, send:Boolean, till_date:datetime.date, send:bool, type:OrderType, comment:str, collect_trades:bool, display_qty:int, broker:str, sender:strб user_data:Object) → Order
      - orders() → OrderIter
      - reset() → True
    - xroad.MdBook
      - asset → Asset

        - data → dict()
        - stop() → None
- xroad.MdFeedStatus
    - asset → Asset
    - data → dict()
- xroad.MdIndicator
    - asset → Asset
    - data → dict()
    - stop() → None
    - send(value: float, type: str) → None
- xroad.MdInfo
    - asset → Asset
    - data → dict()
    - stop() → None
- xroad.MdQuote
    - asset → Asset
    - data → dict()
    - stop() → None
- xroad.MdStat
    - asset → Asset
    - settle_date → datetime.datetime
    - last → dict()
    - quote → dict()
    - open_price → dict()
    - close_price → dict()
    - oi → dict()
    - volume → dict()
    - vwap → dict()
    - min_price → dict()
    - max_price → dict()
    - low_price → dict()
    - high_price → dict()
    - wa_price → dict()
- xroad.MdTrade
    - asset → Asset
    - data → dict()
    - stop() → None
    - send(qty: Integer, price: Float) → None
- xroad.Order
    - asset → Asset
    - name → str
    - active → Boolean
    - side → str
    - price → Float
    - avg_price → Float
    - qty → Float
    - filled_qty → Float
    - state → OrderState
    - type → OrderType
    - exch_id → str
    - exch_ts → datetime.datetime
    - cancel() → None
    - replace(qty: Float, price: Float) → Boolean

        - reset() → None
        - trades() → Iterator
    - xroad.Timer
        - Timer(name::String, event:fn())
        - start(start: Integer|datetime.time, repeat: Integer) → Boolean
        - stop() → True
    - xroad.Props
        - Props(node::String, event:fn(PropsEvent, Prop|PropRow), notify:Boolean) → Props
        - items → Iterator(Prop)
        - query(query:str) → Prop|PropRow|None
        - [prop_name:str] → Prop|None
    - xroad.Prop
        - id → Integer
        - type → PropType
        - name → str
        - value → Any
        - has_link → Boolean
        - rows → Iterator(PropRow)
        - row_count → Integer
        - parent → Prop
        - path → str
        - readonly → Boolean
        - clone → Boolean
        - trace → Boolean
    - xroad.PropRow
        - id → Integer
        - props → Iterator(Prop)
        - parent → Prop
        - delete() → True
        - query(query:str) → Prop|PropRow|None
    - xroad.Speedway
        - Speedway(event:fn(data:str))
        - subscribe(key: string, snapshot: Boolean) → Boolean
        - unsubscribe(key: string) → True

Алго рантайм

Для того чтобы упростить разработку алгоритмов на Python предлагается вместо Python SDK использовать py_algo приложение. py_algo представляет собой Python runtime со встроеными методами и классами, используемыми для разработки торговых алгоритмов. Весь функционал написан на C и при этом доступен из Python кода алгоритма в виде Python классов и методов. При этом, задержка обработки котировки и выставления заявки (tick-to-trade) которую вносит py_algo приложение составляет в среднем 60 микросекунд. Приложение представляет собой консольную утилиту которая инициализирует Python runtime и запускает код алгоритма:

```
~/dev/xroad/release(current) » bin/py_algo -h
usage: py_algo [-h] [-s <name>] [-o <name>] ...
   -h - print this help
   -s <name> - run as stand alone process with access to entire system features
   -o <name> - run as out-of-system process without access to entire system. Best mode is you want to access only cache
   -S - print logs to stout
   -a - autostart
   -c - check all assets state (feed and trading) before order send/replace
   -p <file> - python code to execute
   -g <string expression> - pass string expression to python code as variable xroad.args
```

Для того, чтобы запустить алгоритм (например реализованный в файле algo.py) достаточно выполнить команду:

```
bin/py_algo -S -s algo1 -a -p `pwd`/algo.py
```

при этом algo1 это название сконфигурированной ноды. Помимо ручного запуска алгоритма, есть возможность автоматического запуска алгоритма как сервис.

Справочное руководство Python Algo

Перечисления

xroad.TimeInForce

время жизни заявки

- day = 48 - заявка валидна в течении дня
- GTC = 49 - заявка валидна по ее не отмнят
- open = 50 - заявка валидна только в течении сессии открытия
- IOC = 51 - заявка исполняется немедленно (частично или полностью) или отклоняется.
- FOK = 52 - заявка исполняется немедленно полностью или снимается
- GTD = 54 - заявка валидна до определенной даты
- close = 55 - заявка валидна в течении сессии закрытия

xroad.OrderFlag

конфигурационные настройки заявки

- as_mmaker = 1 - заявка подается от лица официального маркер мейкера
- participate_only = 2
- close_position = 16
- autodelete = 256
- ignore_dyn_limits = 1024
- test = 2048
- reduce_only = 4096

- otc = 8192 - OTC заявка
- reinstate_on_fail = 16384
- cancel_on_cross = 32768
- dont_crossmm = 65536
- ignore_mdata = 131072
- quote = 262144

xroad.OrderEvent

события времени жизни заявки

- activated = 1 - заявка активирована на бирже
- canceled = 2 - заявка снята
- trade = 3 - заявка исполнилась полностью или частично
- rejected = 4 - заявка была отклонена биржей
- cancel_rejected = 5 - операция отмены заявки была отклонена биржей
- replaced = 6 - операция изменения параметров завяки была успешно обработана биржей
- replace_rejected = 7 - операция изменения параметров заявки была отклонена биржей

xroad.AssetEvent

события возникающие у объекта типа Asset

- pos_changed = 1 - событие об изменении позиции (в случае купли или продажи)
- error = 2 - возникло ошибочние событие (например заявка была отклонена биржей)
- info = 3 - информационное событие (например, заявка исполнилась)

xroad.EnvEvent

события уровня приложения

- reconfig = 1 - конфигурация изменилась
- start = 2 - приложение получило start событие
- stop = 3 - приложение получило stop событие
- reset = 4 - приложение получило reset событие
- msg = 5 - приложение получило сообщение от другого приложения

xroad.TradingStatus

торговый интервал инструмента

- regular_trading = 1 - торгуется в основную торговую сессию
- opening_period = 2 - торгуется в пред-торгорвый период
- closing_period = 3 - торгуется в период закрытия
- opening_auction = 4 - торгуется в аукцион открытия
- closing_auction = 5 - торгуется в аукцион закрытия
- auction = 6 - торгуется в аукцион
- auction_price_trading = 7 - период определения цены аукциона
- halt = 8 - инструмент не торгуется

- stopped = 9 - торги остановлены
- auction_closing = 10 - торгуется в аукцион закрытия
- unknown = 11 - неопределенный торговый интервал

xroad.MdataFeedStatus

статус протока данных

- offline = 0 - поток данных не работает
- online = 1 - поток данных работает
- unknown = 2 - неизвестный статус

xroad.OrderState

статус заявки

- initial = 0 - начальный статус заявки
- active = 1 - заявка активна
- destroyed = 2 - заявка удалена
- canceled = 3 - заявка отменена
- rejected = 4 - заявка отклонена
- filled = 5 - заявка полнстью исполнена
- expired = 6 - срок действия заявки истек
- awaiting_active = 7 - заявка ожидает актитвации
- awaiting_destroy = 9 - заявка ожидает удаления
- awaiting_cancel = 9 - заявка ожидает отменв
- awaiting_replace = 10 - заявка ожидает изменения

xroad.OrderType

тип заявки

- limit = 49 - лимитная
- market = 50 - рыночная

xroad.PropsEvent

событие изменения свойства

- changed = 1 - приложение получило сообщение об изменении свойства
- row_added = 2 - приложение получило сообщение об добавлении новой строки в таблице
- row_deleted = 3 - приложение получило сообщение об удалении строки из таблицы

xroad.PropType

тип свойсва

- string = 1 - свойство имеет строковый тип

- boolean = 2 - свойство имеет булевый тип
- integer = 3 - свойство имеет integer тип
- double = 4 - свойство имеет double тип
- time = 5 - свойство имеет time тип
- table = 6 - свойство имеет table тип
- date = 7 - свойство имеет date тип

Глобальные функции

xroad.set_env_event(callback:fn) → None

устанавливает функцию, которая будет вызываться в случае возникновения события уровня приложения (см. EnvEvent)

Параметры:

- callback - функция типа function(EnvEvent, dict)

Пример:

```
def env_events(event, data):
   if event == EnvEvent.reconfig:
      # reload config
      print(get_config())
   elif event == EnvEvent.start:
      # e.g. start algo
   elif event == EnvEvent.stop:
      # e.g. stop algo
   elif event == EnvEvent.msg:
      print(data)
   elif event == EnvEvent.prop:
      # property changed. data is dict {"name": "prop_name", "value": "prop_value"}


set_env_event(env_events)
```

xroad.send_msg(node: str|int, data::dict)

посылает сообщение другой ноде системы

Параметры:

- node - имя ноды или ее id

- data - отправляемые данные

Пример:

```
send_msg(node="algo2", data={"parameter1": 10, "parameter2": "some usefull info"})
```

xroad,md_hbt_start() → None

в случае, если алгоритм является источником данных он должден посылать хардбит

xroad.md_hbt_stop() → None

наоборот выключает хардбит

xroad.start() → None

переводит ноду в состояние active

xroad.stop() → None

переводит ноду в состояние inactive и снимает все активные заявки. Новые заявки посылаться не будут

xroad.exit() → None

корректно завершает работу py_algo

xroad.reconfig() → None

производит реконфигурацию окружения. В частности, перечитывает инструменты

Классы

xroad.Asset

определяет операции уровня инструмент

Asset(name::String, event:fn(Asset, AssetEvent, dict)) → Asset

конструктор создает объект типа Asset

Параметры:

- name - имя инструмента
- event - необязательная вызываемая функция для обработки событий типа AssetEvent
- Возвращаемое значение:
  - объект типа Asset или генерируется исключение, в случае если инструмент с таким именем не найден

Пример:

```
import logging
from xroad import *

logger = logging.getLogger()

a1 = Asset(name="LKOH.TQBR")

def asset_event(event, data):
   if event == AssetEvent.pos_changed:
      logging.info(data)

a2 = Asset(name="SBER.TQBR", event=asset_event)
```

name → str

имя инструмента

total_buy → Float

сколько всего куплено

avg_buy → Float

средняя цена покупки

total_sell → Float

сколько всего продано

avg_sell → Float

средняя цена продажи

md_stat → MdStat

рыночная информация инструмента (см. MdStat)

trading_status → TradingStatus

статус инструмента

ref → dict

параметры инструмента:

- lot_size: Integer - размер лота
- currency: str - валюта инструмента
- alias: str - xroad название инструмента
- name: str - биржевое имя инструмента
- exch_id: Integer - биржевой ID инструмента (если он есть)
- board: str - биржевой подраздел инструмента (если он есть)
- exchange: str - название биржи
- cfi: str - CFI код инструмента
- isin: str - ISIN инструмента (если он есть)
- face_value: Float - номинальная стоимость (для облигаций)
- exp_datetime: datetime.datetime - дата экспирации
- callput: str - для опционов ('call' или 'put')
- strike: Float - страйк цена для опционов
- contract_size: Integer - размер контракта для фьючерсов
- margin_short: Integer - ГО продавца
- margin_long: Integer - ГО покупателя
- dividend_date: datetime.datetime - дата выплаты дивидетнов (для акций)
- dividend_value: Float - размер выплачиваемых дивидентов
- tick_info: dict() - информация по шагу цены. Пример:

```
[{'price_min': 0, 'price_max': 1000000, 'tick_size': 0.01, 'tick_value': 0.01, 'precision': 2}, ...]
```

md_quote(event::fn(quote:MdQuote), snapshot:Boolean, source:str) → MdQuote|None

создает подписку на квоту

Параметры:

- event - вызываемая функция типа fn(MdQuote)
- snapshot - необязательный параметр, который определяет нужно ли получить снэпшот данных в начале

- source - необязательный параметр, указывающий на источник квоты (имя ноды)

Возвращаемое значение:

- объект типа MdQuote или None

Пример:

```
import logging
from xroad import *

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

def on_quote(quote):
   logging.info("asset = {}, quote = {}".format(quote.asset.name, quote.data))

a.md_quote(event=on_quote)
```

md_info(event::fn(info:MdInfo), snapshot:Boolean, source:str) → MdInfo|None

подписывается на различную статистику по инструменту (например: OI, цена последней сделки, и т.д.

Параметры:

- event - вызываемая функция типа fn(MdInfo)
- snapshot - необязательный параметр, который определяет нужно ли получить снепшот данных в начале
- source - необязательный параметр, указывающий на источник данных (имя ноды)

Возвращаемое значение:

- объект типа MdInfo или None

Пример:

```
import logging
from xroad import *

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

def on_info(info):
   logging.info("asset = {}, info = {}".format(info.asset.name, info.data))

a.md_info(event=on_info)
```

md_book(event::fn(book:MdBook), snapshot::Boolean, source:str) → MdBook|None

подписывается на L2 данные по инструменту

Параметры:

- event - вызываемая функция типа fn(MdBook)
- snapshot - необязательный параметр, который определяет нужно ли получить снепшот данных в начале
- source - необязательный параметр, указывающий на источник данных (имя ноды)

Возвращаемое значение:

- объект типа MdBook или None

Пример:

```
import loggingfrom xroad import *

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

def on_book(book):
   logging.info("asset = {}, book = {}".format(book.asset.name, book.data))

b = a.md_book(event=on_book)
```

md_trade(event::fn(trade:MdTrade), snapshot::Boolean, source:str) → MdTrade|None

подписывается на рыночные сделки по инструменту

Параметры:

- event - вызываемая функция типа fn(MdTrade)
- snapshot - необязательный параметр, который определяет нужно ли получить снепшот данных в начале
- source - необязательный параметр, указывающий на источник данных (имя ноды)

Возвращаемое значение:

- объект типа MdTrade или None

Пример:

```
import logging
from xroad import *

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

def on_trade(trade):
   logging.info("asset = {}, trade = {}".format(trade.asset.name, trade.data))

t = a.md_trade(event=on_trade)
```

md_indicator(event::fn(trade:MdIndicator), snapshot::Boolean, source:str) → MdIndicator|None

подписывается на значение индикатора инструмента

Параметры:

- event - вызываемая функция типа fn(MdIndicator)
- snapshot - необязательный параметр, который определяет нужно ли получить снепшот данных в начале
- source - необязательный параметр, указывающий на источник данных (имя ноды)

Возвращаемое значение:

- объект типа MdIndicator или None

Пример:

```
import logging
from xroad import *

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

def on_indicator(indi):
   logging.info("asset = {}, indi = {}".format(indi.asset.name, indi.data))

t = a.md_indicator(event=on_indicator)
```

md_feed_status(event::fn(FeedStatus), snapshot::Boolean) → MdFeedStatus|None

подписывается на статус протока данных

Параметры:

- event - вызываемая функция типа fn(MdFeedStatus)

Возвращаемое значение:

- объект типа MdFeedStatus или None

md_stop() → None

удалает все подписки

cancel() → True

отменяет все активные заявки

buy(name:str, account:str, client_code:str, qty:float, price:float, tif:TimeInForce, portfolio:str, flags:int, event:fn(OrderEvent, dict), till_date:datetime, send:Boolean, till_date:datetime.date, send:bool, type:OrderType, comment:str, collect_trades:bool, display_qty:int, broker:str, sender:str, user_data:Object) → Order

создает заявку на покупку

Параметры:

- name - необязательный. Имя ордера, должно быть уникальным

- account - необязательный. Счет клиента
- client_code - необязательный. Код клиента
- qty - необязательный. Кол-во заявки в лотах
- price - необязательный. Цена заявки
- tif - см. TimeInForce enum
- portfolio - необязательный. Книга трейдера
- flags - необязательный. Настройки заявки (см. OrderFlag enum)
- event - необязательный. Вызываемая функция для полного контроля поведения завяки
- till_date - необязательный. Время жизни заявки (используется в том случае, если tif = TimeInForce.GTD)
- send - необязательный (default: True). По умолчанию заявка посылается на биржу в момент создания. Этот параметр позволяет выключить данное поведение
- comment - необязательный. Комментарий в заявке
- collect_trades - необязательный (default: False). Позволяет сохранять сделки в объекте Order, чтобы впоследствии их просмативать итератором
- display_qty - необязательный. Видимое кол-во в заявке
- broker - необязательный. Брокер, используется для маршрутизации заявки
- sender - строковый идентификатор лица/алгоритма от имени которого посылается заявка
- user_data - необязательный. Пользовательские данные, которые должны быть ассоциированы с ордером

Возвращаемое значение:

- объект типа Order
- исключение в случае ошибки

```
import logging
from xroad import *

cfg = {"account": "acc1", "client_code": "ccode1", "tif": TimeInForce.day}

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

b = a.buy(qty=1, price=256.12, **cfg);
```

sell(name:str, account:str, client_code:str, qty:float, price:float, tif:TimeInForce, portfolio:str, flags:int, event:fn(OrderEvent, dict), till_date:datetime, send:Boolean, till_date:datetime.date, send:bool, type:OrderType, comment:str, collect_trades:bool, display_qty:int, broker:str, sender:strб user_data:Object) → Order

создает заявку на продажу

Параметры:

- name - необязательный. Имя ордера, должно быть уникальным
- account - необязательный. Счет клиента
- client_code - необязательный. Код клиента
- qty - необязательный. Кол-во заявки в лотах
- price - необязательный. Цена заявки
- tif - см. TimeInForce enum

- portfolio - необязательный. Книга трейдера
- flags - необязательный. Настройки заявки (см. OrderFlag enum)
- event - необязательный. Вызываемая функция для полного контроля поведения завяки
- till_date - необязательный. Время жизни заявки (используется в том случае, если tif = TimeInForce.GTD)
- send - необязательный (default: True). По умолчанию заявка посылается на биржу в момент создания. Этот параметр позволяет выключить данное поведение
- comment - необязательный. Комментарий в заявке
- collect_trades - необязательный (default: False). Позволяет сохранять сделки в объекте Order, чтобы впоследствии их просмативать итератором
- display_qty - необязательный. Видимое кол-во в заявке
- broker - необязательный. Брокер, используется для маршрутизации заявки
- sender - строковый идентификатор лица/алгоритма от имени которого посылается заявка
- user_data - необязательный. Пользовательские данные, которые должны быть ассоциированы с ордером

Возвращаемое значение:

- объект типа Order
- исключение в случае ошибки

```
import logging
from xroad import *

cfg = {"account": "acc1", "client_code": "ccode1", "tif": TimeInForce.day}

logger = logging.getLogger()

a = Asset("LKOH.TQBR")

b = a.sell(qty=1, price=256.12, **cfg);
```

orders() → OrderIter

создает итератор для получения списка ордеров

reset() → True

сбрасывает статистику инструмента (avg_buy, avg_sell, total_buy, total_sell)

xroad.MdBook

возвращаемое значение подписки на книги (см. Asset.md_book)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

биржевые данные, книга ордеров. Пример:

```
{'bids': [{'price': 100.1, 'qty': 1}, {'price': 100, 'qty': 2}, ...], 'asks': [...]}
```

stop() → None

останавливает подписку

xroad.MdFeedStatus

возвращаемое значение подписки на статус потока данных (см. Asset.md_feed_status)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

последнее значение. Пример:

```
{'ts': datetime.datetime, 'status': MdataFeedStatus.offline}
```

xroad.MdIndicator

возвращаемое значение подписки на индикатор (см. Asset.md_indicator)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

последнее значение. Пример:

```
{'ts': datetime.datetime, 'type': 'INDI1', 'value': 1000}
```

stop() → None

останавливает подписку

send(value: float, type: str) → None

отправить расчетное значение индикатора (type)

Параметры:

- value: Float - значение индикатора
- type: str - название индикатора

xroad.MdInfo

возвращаемое значение подписки на биржевую информацию (см. Asset.md_info)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

последние данные. Пример:

```
{'oi': .., 'min': .., 'max': .., 'open': .., 'close': .., 'high': .., 'low': .., 'last': ...}
```

stop() → None

останавливает подписку

xroad.MdQuote

возвращаемое значение подписки на квоту (см. Asset.md_quote)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

последняя квота. Пример:

```
{'bid': {'price': 100.1, 'qty': 100}, 'ask': {'price': 100.2, 'qty': 200}, 'ts': datetime.datetime}
```

stop() → None

останавливает подписку

xroad.MdStat

рыночная информация по инструменту возвращаемая Asset.md_stat

asset → Asset

возвращает ссылку на объект Asset

settle_date → datetime.datetime

дата поставки

last → dict()

последняя сделка

```
{'price': 100.1, 'qty': 100, 'ts': datetime.datetime}
```

quote → dict()

последняя квота

```
{'bid_price': 100.1, 'bid_qty': 10, 'ask_price': 100.2, 'ask_qty': 20, 'ts': datetime.datetime}
```

open_price → dict()

цена открытия

```
{'value': 100.1, 'ts': datetime.datetime}
```

close_price → dict()

цена закрытия

```
{'value': 100.1, 'ts': datetime.datetime}
```

oi → dict()

открытый интерес

```
{'value': 100.1, 'ts': datetime.datetime}
```

volume → dict()

объем

```
{'value': 100.1, 'ts': datetime.datetime}
```

vwap → dict()

VWAP цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

min_price → dict()

минимально допустимая цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

max_price → dict()

максимально допустимая цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

low_price → dict()

максимальная цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

high_price → dict()

минимальная цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

wa_price → dict()

средневзвешенная цена

```
{'value': 100.1, 'ts': datetime.datetime}
```

xroad.MdTrade

возвращаемое значение подписки на сделку (см. Asset.md_trade)

asset → Asset

возвращает ссылку на объект Asset

data → dict()

последняя сделка. Пример:

```
{'price': 100.1, 'qty': 10, 'ts': datetime.datetime}
```

stop() → None

останавливает подписку

send(qty: Integer, price: Float) → None

позволяет сгенерировать сделку по инструменту

Параметры:

- qty: Integer - количество в сделке
- price: Float - цена сделки

xroad.Order

ордер, возвращаемое значение Asset.buy(sell)

asset → Asset

возвращает ссылку на объект Asset

name → str

уникальное имя ордера

active → Boolean

возвращает True если ордер активный, False - ордер выполнен или отменен

side → str

'B' - ордер на покупку, 'S' - на продажу

price → Float

цена ордера

avg_price → Float

средневзвешенная цена ордера

qty → Float

количество в ордере

filled_qty → Float

исполненное количество в ордере

state → OrderState

Стейт ордера

type → OrderType

тип заявки

exch_id → str

биржевой ID заявки

exch_ts → datetime.datetime

биржевое время заявки

cancel() → None

отменить заявку

replace(qty: Float, price: Float) → Boolean

заменить завявку

Параметры:

- qty: Float - новое кол-во в заявке
- price: Float - новая цена заявки

reset() → None

сбрасывает статистику исполнения (avg_price, filled_qty, trades)

trades() → Iterator

возвращает итератор, который позволяет просмотреть все сделки ордера

xroad.Timer

таймер

Timer(name::String, event:fn())

конструктор создает объект типа Timer

Параметры:

- name - имя таймера
- event - вызываемая функция

start(start: Integer|datetime.time, repeat: Integer) → Boolean

запускает таймер

Параметры:

- start: Integer|datetime.time - время срабатывания таймера или через N микросекунд или в конкретное время
- repeat: Integer - повторить таймер через N микросекунд

stop() → True

остановить таймер

xroad.Props

работа со свойствами

Props(node::String, event:fn(PropsEvent, Prop|PropRow), notify:Boolean) → Props

конструктор создает объект типа Props

Параметры:

- node: String - имя ноды со свойствами. Если имя коды не заданы, то будут использоваться свойства текущей воды
- event: fn - калбэк функция, которая вызывается если произошли изменения в свойствах текущей ноды
- notify: Boolean - при изменении свойств, ноде владельцу измененного свойства будет отправлено уведомление

items → Iterator(Prop)

создает итератор для перебора свойств

query(query:str) → Prop|PropRow|None

позволяет искать свойства. Пример:

```
props = Props()
p1 = props.query(query='prop1')
p2 = props.query(query='algo[id=1]/instrument')
p3 = props.query(query='algo[instrument=SBER.TQBR]/bid_price')
```

[prop_name:str] → Prop|None

получить свойство по имени. Пример:

```
props = Props(node='algo')
p1 = props['broker']
p2 = props['client_code']
```

xroad.Prop

класс свойства

id → Integer

ID свойства

type → PropType

тип свойства

name → str

имя свойства

value → Any

значение свойства. Пример:

```
props = Props()
p = props['broker']
print(p.value)
```

has_link → Boolean

свойства слинковано с другим свойством

rows → Iterator(PropRow)

если свойство имеет табачный тип, то данное свойство позволяет перебрать все строки этой таблицы. Пример:

```
props = Props()
algos = props['algo']
for algo in algos.rows:
    print(algo.id)
```

row_count → Integer

количество строк в таблице

parent → Prop

возвращает родительское свойство

path → str

возвращает полный путь свойства

readonly → Boolean

возваращает значение readonly атрибута

clone → Boolean

возваращает значение clone атрибута

trace → Boolean

возваращает значение trace атрибута

xroad.PropRow

класс строки в табачном свойстве

id → Integer

ID свойства

props → Iterator(Prop)

итератор, позволяющий итерироваться по свойствам строки. Пример:

```
# print all prop names in rows
props = Props()
algos = props['algo']
for algo in algos.rows:
    for prop in algo.props:
        print(prop.name)
```

parent → Prop

возвращает родительское свойство

delete() → True

удаляет текущую строку

query(query:str) → Prop|PropRow|None

позволяет искать свойства (см. Props.query)

xroad.Speedway

получение данных из speedway подсистемы

Speedway(event:fn(data:str))

конструктор создает объект типа Speedway

Параметры:

- event - вызываемая функция

subscribe(key: string, snapshot: Boolean) → Boolean

подписывается на данные по ключу

Параметры:

- key: string - ключ данных
- snapshot: boolean - подписка на снэпшот

unsubscribe(key: string) → True

отписаться от данных