Tibet策略开发说明

```
History:
Ver:  0.1  Date: 2018/7/12  Author: zhangbin
    1. 创建
```

1. 简介

`Tibet`程序自动化交易系统。

系统采用分布式、模块化设计架构，每个模块均可作为独立的服务单独进行部署运行。

支持`期货CTP`和`股票XTP`自动化交易接口。

2. 框架说明

2.1 程序结构

策略项目称之为策略运行器`StrategyRunner`，每个`Runner`是单独的进程，并加载运行一个策略。

项目文件结构：

```
Runner/
    ├── data
    ├── etc
    |   └── settings.yaml        系统配置
    ├── logs                         日志目录
    ├── run
    ├── src
    |   ├── server-strategy-runner.py   运行入口程序
    |   ├── strategies           策略目录
    ├── temp
    └── tests
```

运行参数

```
Runner.py command [ options] strategy_name

Command:
   list        - 列出指定用户id下所有策略信息
   create      - 创建策略项目
   remove      — 删除策略项目
   pull        - 从仓库中下载策略到本地目录
   upload      - 本地策略上传更新到仓库
   run         - 运行本地策略
Options:
    --user        - 用户名
    --password    - 访问口令
    --remote      - 执行远程策略，自动从仓库下载策略到本地再运行
    --launcher_id  - 启动策略Runner的加载器标识

Examples:
    Runner.py run --user=scott demo  运行本地 "demo"策略
    Runner.py run --remote S1000011  加载仓库策略 "S1000011“到本地运行
```

策略配置

`src/strategies/` 是策略宿主目录。`create`，`pull`，`run` 等命令将在此目录中创建、更新、运行指定的策略模块。

策略模块结构

```
 demo/
    ├── __init__.py 必须存在，视为模块
    ├── config.yaml 策略运行配置参数
    ├── main.py     策略运行入口模块
```

config.yaml

```
# 策略的运行配置参数
name: 'demo'                模块名称
user: 'scott'               交易用户名称
quotas:                     分配的资金账号
  - name:     'q001'        配额名称
    account:  'htqh-01'     资金账户名称
    product:  'future'      期货、股票 (future / stock )
    limit:    2000000       资金最大配额
strategy:
  id: ''
  configs:                  此处由开发人员自行配置
    sub_ticks: 'AP810,RM809'   订阅指定的行情Tick
    sub_bar_1m: 'AP810,RM809'
    sub_bar_5m: 'AP810,RM809'
```

2.2 Runner系统框架

2.2.1. 主要对象

- Controller
- Handler (Futures , Stocks)
- QuotaAccount
- Context
- StrategyBean

@ Controller

核心控制对象

.setTimer(action=None,timeout=1)

启动定时器， `action` 指定触发事件函数 , 默认触发 `onTimer()` 。返回定时器对象。

action原型:

```
def onTimer(timer,ctx):
    do someting...
```

.killTimer(timer)

停止定时器

@Handler

`controller` 中包含不同的处理器 `handler`. 不同的 `handler` 表示一种金融产品的接入，目前支持: 股票 `stockHandler` 和 期货 `futureHandler` 。

资源对象

.subTicks(symbol,gateway=CTP)

行情订阅方法。

参数:

- `symbol` - 合约名称
- `gateway` - 接入类型，默认采用 CTP 接口

.subBars(symbol,scale,gateway=CTP)

行情订阅方法。

参数:

- `symbol` - 合约名称
- `scale` - 时间刻度, "1m,2m,5m,15m,30m,1h,1d"
- `gateway` - 接入类型，默认采用 CTP 接口

.buy(symbol,price,volume,priceType=LIMITPRICE,account='')

期货买开

参数:

- `symbol` - 合约名称
- `price` - 价格
- `volume` - 数量
- `priceType` - 价格类型 , 默认: 限价
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.sell(symbol,price,volume,priceType=LIMITPRICE,account='')

期货卖平

参数:

- `symbol` - 合约名称
- `price` - 价格
- `volume` - 数量
- `priceType` - 价格类型 , 默认: 限价
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.short(symbol,price,volume,priceType=LIMITPRICE,account='')

期货卖开

参数:

- `symbol` - 合约名称
- `price` - 价格
- `volume` - 数量
- `priceType` - 价格类型 , 默认: 限价
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.cover(symbol,price,volume,priceType=LIMITPRICE,account='')

期货买平

参数:

- `symbol` - 合约名称
- `price` - 价格
- `volume` - 数量
- `priceType` - 价格类型 , 默认: 限价
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

`buy,sell,short,cover` 调用操作均返回委托订单编号数组.
`开仓操作一般只返回一个订单编号，平仓操作（平今、平昨）会产生多个委托订单编号（拆单）`

例如:

```
['CTP.1','CTP.2',...]
```

.cancelOrder(order_id,account='')

期货撤单

参数:

- `order_id` - 订单编号
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.cancelAllOrders(account='')

期货撤除全部委托单

参数:

- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getOrder(order_id,account='')

期货查询委托单详情

参数:

- `order_id` - 委托订单编号
- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getAllOrders(account='')

期货查询所有委托单详情

参数:

- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getAllTrades(account='')

期货查询所有交易

参数:

- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getAllPositions(account='')

期货查询当前所有持仓情况

参数:

- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getCurrentAccount(account='')

期货查询当前账户资金明细

参数:

- `account` - 资金账户名， 默认： 第一个配额账户(quotaAccount)

.getContract(symbol)

期货查询指定合约明细

参数:

- `symbol` - 合约名称

.getLastestTick(symbol)

期货查询合约最新的行情Tick

参数:

- `symbol` - 合约名称

.loadTicks(symbol,limit=0)

期货 加载合约历史行情Tick记录

参数:

- `symbol` - 合约名称
- `limit` - 返回记录数 , 默认: 0 ,不限数量

.loadBars(symbol,scale,limit=0)

期货 加载合约历史行情Bar记录

参数:

- `symbol` - 合约名称

- `scale` - 时间刻度 , 1m,2m,5m,15m,30m,1h,1d
- `limit` - 返回记录数 , 默认: 0 ,不限数量

@QuotaAccount

资金配额账户

@Context

上下文对象context是策略框架与用户程序上下文交换的对象。 `context` 包含了策略开发过程中所必备工具和资源。

两种方式获得 `context` 对象

- 在策略代码 `main.py` 文件中声明变量 `context = None` ，框架在启动策略时，自动初始化context变量。
- 在策略框架定义的诸多回调事件中，均带有上下文对象 `ctx` ,默认最后一个参数。

```
    init(ctx):
    onTick(tick,ctx):
    onTrade(trade,ctx):
```

资源对象

.controller

控制器对象

.props

属性集合（dict类型）。 用户代码可以将数据存入 `props` 对象，便于在不同事件、状态切换时得到便捷的访问和保持。

```
# src/strategies/demo/main.py

def init(ctx):
    ctx.props['value'] = 100. # 存入自定义值

def onTick(tick,ctx):
    ctx.props['last_tick'] = tick    # 保存最新的行情tick

def onTimer(timer,ctx):
    print ctx.props['last_tick']     # 取出保存的行情记录
```

.tools

工具类对象，提供多种工具类库引用，例如 TA-Lib的 `tools.ArrayManager` 。

.logger

日志输出对象，分别提供 `.debug()` , `info()` , `error()` , `warn()` 等日志输出方法。

.future

期货访问接口对象

.stock

股票访问接口对象

.configs

当前策略配置文件 `config.yaml` 中定义的自定义参数 `configs`

```
strategy:
id: ''
configs:      # 此处由开发人员自行配置
    sub_ticks: 'AP810,RM809'
    sub_bar_1m: 'AP810,RM809'
    sub_bar_5m: 'AP810,RM809'
```

.mongodb

数据库连接对象，提供用户程序直接访问数据存储服务的接口。

@StrategyBean

策略程序作为单独的python程序模块部署在 `src/strategies/` 内。

创建策略:

```
Runner.py create MyStrategy
```

程序将在 `strategies` 目录中自动创建 `MyStrategy` 模块，文件包含:

- `__init__.py`
- `main.py`

- `config.yaml`

配置策略参数

```
name: 'demo'
user: 'scott'           # 交易用户名称
quotas:                 # 分配的资金账号
  - name:   'q001'      # 配额名称
    account: 'htqh-01' # 资金账户名称
    product: 'future'   # 期货、股票
    limit:  2000000     # 资金最大配额
strategy:
  id: ''
  configs:      # 此处由开发人员自行配置
    sub_ticks: 'AP810,RM809'
```

添加资金配额账户到条目 `quotas` 下，这里配置了资金账户名 `htqh-01` ,类型为 `future` 期货的配额账户，限定资金最大为 `200000` .

开发策略代码 (main.py)

一个策略需要定义若干的事件函数，这些函数基于事件驱动被调用。主要包含：

init(ctx)

策略的初始化

open(ctx)

策略的开始运行

onTimer(timer,ctx)

定时器触发

onTick(tick,ctx)

订阅的行情Tick数据到达

onBar(bar,ctx)

订阅的行情Bar数据到达，`bar.scale , bar.product` 用于区分不同合约产品和时间刻度

onTrade(trade,ctx)

交易事件到达

- trade - vnpy.trader.vtObject.VtTradeData

onOrder(order,ctx)

委托事件到达

- order - vnpy.trader.vtObject.VtOrderData