BacktestingEngine 回测引擎

eventEngine=EventEngine()	#事件引擎
strategyEngine	#引用策略引擎
listDataHistory	#Tick历史数据列表
dictOrder	#限价单字典
currentData	#最新Tick数据
listTrade	#回测的成交字典
orderRef	#报单编号
tradeID	#成交编号
Id	#回测编号
setStrategyEngine	#设置策略引擎
connectMongo	#连接MongoDb
loadMongoDataHistory	#从MongoDb获取Tick历史
connectMysql	#连接MysqlDb
loadMysqlDataHistory	#从MysqlDb获取Tick历史
getMysqlDeltaDate	#从MysqlDB获取交易日天数差
processLimitOrder	#仿真处理限价单
executeLimitOrder	#仿真限价单成交处理
startBacktesting	#开始回测
sendOrder	#回测发单
cancelOrder	#回测撤单
subscribe	#仿真订单合约
selectInstrument	#读取合约数据
saveTradeData	#保存交易记录
saveTradeDataToMysql	#保存交易记录到MysqlDb
writeLog	#写日志

strategyEngine 策略引擎

__eventEngine	#引用事件引擎
mainEngine	#主引擎
backtesting	#是否回测
today	#当日
__dictOrder	#保存所有报单的字典
dictStrategy	#保存策略的字典
__dictSymbolStrategy	#合约代码,策略对象映射
__dictOrderRefStrategy	#报单编号,策略对象映射
__dictStopOrder	#保存合约与止损单映射字典
__mysqlConnected	#数据库是否连接
__mysqlConnection	#数据库连接
__connectMysql	#连接mysqldb
__recordTickToMysql	#保存tick至mysqldb
__processStopOrder	#处理停止单
__registerEvent	#注册事件监听
loadTickFromMysq l	#从mysql加载tick历史
getMysqlDeltaDate	#从mysql获取交易日差
updateMarketData	#更新行情数据
updateOrder	#事件响应，报单更新
updateTrade	#时间响应，成交更新
sendOrder	#发单
cancelOrder	#撤单
createStrategy	#创建策略（实例化）
registerStrategy	#注册合约代码与策略映射
placeStopOrder	#下止损单
cancelStopOrder	#撤销止损单
startAll	#启动所有策略
stopAll	#停止所有策略
writeLog	#写日志

MainEngine(DemoEngine.py) 主引擎

ee=EventEngine()	#事件引擎
md=DemoMdApi(ee)	
td=DemoTdApi(ee)	
countGet	#查询延时计数
lastGet	#上次查询的性质
login	#登录
initGet	#初始化查询
subscribe	#订阅合约
getAccount	#查询账户
getInvestor	#查询投资者
getPosition	#查询持仓
getInstrument	#获取合约
getAccountPosition	#循环查询账户和持仓
sendOrder	#发单（委托下单）
cancelOrder	#撤单
insertInstrument	#插入合约对象
selectInstrument	#获取合约信息对象
saveInstrument	#保存合约信息
exit	#退出

EventEngine 事件引擎

__queue	#事件队列
__active	#事件引擎开关
__thread	#事件处理线程
__timer	#计时器
__handlers	#事件、调用函数 映射字典
__init	#初始化事件
__run	#引擎运行
__process	#处理事件
__onTimer	#向事件队列存入计数器事件
start	#引擎启动
stop	#停止引擎
register	#注册事件处理函数监听
unregister	#注销事件处理函数监听
put	#事件队列

Event 事件类

type	#事件类型
dict	#事件数据字典

StrategyTemplate策略模板（基类）

name	#策略名称
symbol	#策略交易的合约
engine	#策略引擎对象
trading	#策略是否启动交易
onTick	#行情更新事件
onTrade	#交易更新事件
onOrder	#报单更新事件
onStopOrder	#止损单更新事件
onBar	#K线数据更新
start	#启动交易
stop	#停止交易
loadSetting	#载入（参数）设置
buy	#买入开仓
cover	#买入平仓
sell	#卖出平仓
short	#卖出开仓

demoStrategy 策略

name	#策略名称
symbol	#策略交易的合约
engine	#策略引擎对象
trading	#策略是否启动交易
pos	#持仓头寸
onTick	#行情更新事件
onTrade	#交易更新事件
onOrder	#报单更新事件
onStopOrder	#止损单更新事件
onBar	#K线数据更新
start	#启动交易
stop	#停止交易
loadSetting	#载入（参数）设置
buy	#买入开仓
cover	#买入平仓
sell	#卖出平仓
short	#卖出开仓
initStrategy	#初始化
strToTime	#字符串时间转换

DemoMdApi 行情接口

-memberName
-memberName

DemoTdApi 交易接口

-memberName
-memberName

Tick 分笔数据

-memberName
-memberName

Trade 成交单

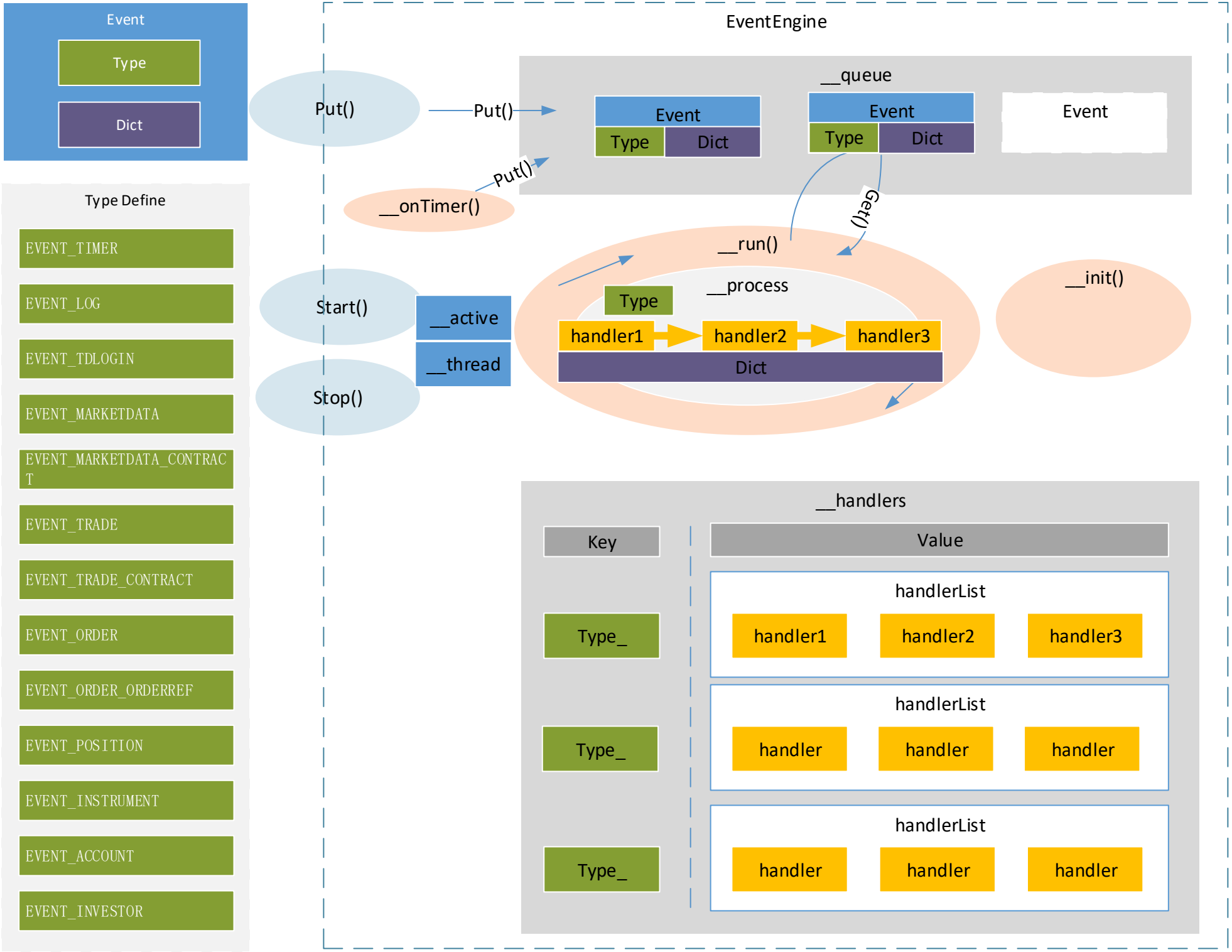
-memberName
-memberName

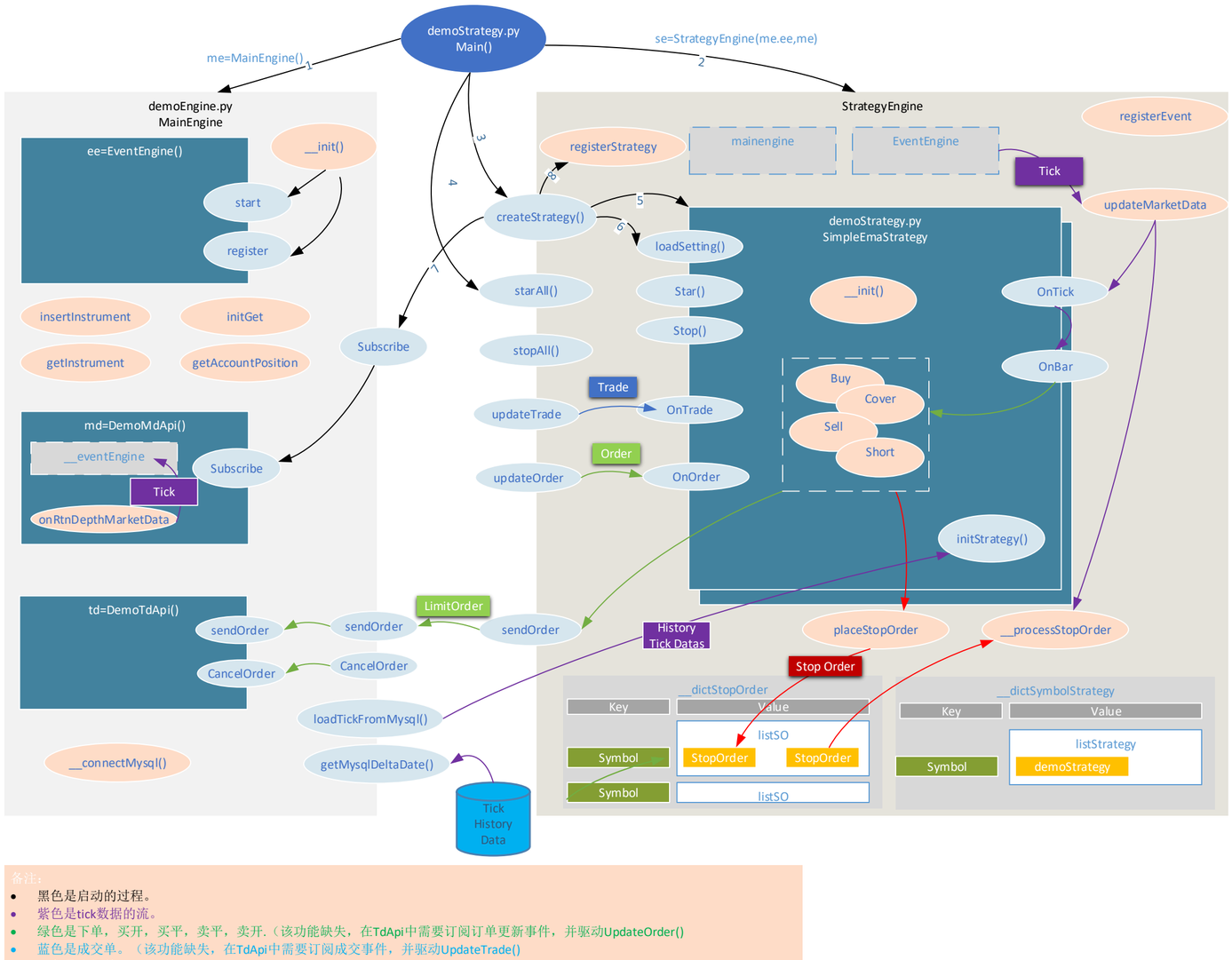
Order 发单

-memberName
-memberName

stopOrder 止损单

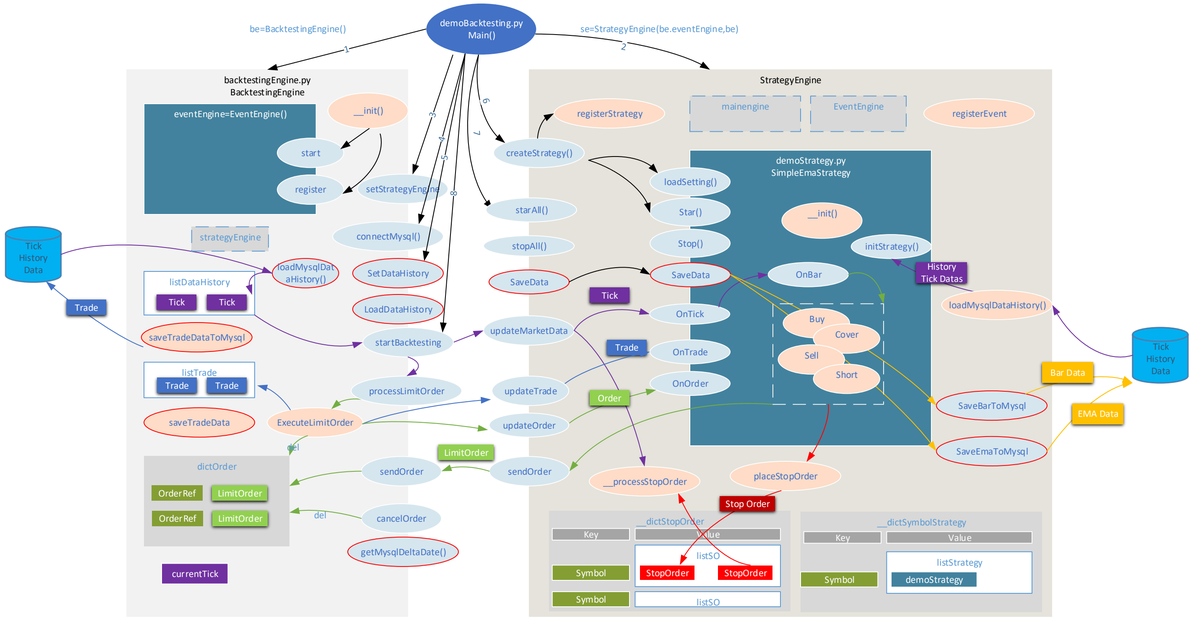
-memberName
-memberName


Event
Type
Dict
Type Define
EVENT_TIMER
EVENT_LOG
EVENT_TDLOGIN
EVENT_MARKETDATA
EVENT_MARKETDATA_CONTRACT
EVENT_TRADE
EVENT_TRADE_CONTRACT
EVENT_ORDER
EVENT_ORDER_ORDERREF
EVENT_POSITION
EVENT_INSTRUMENT
EVENT_ACCOUNT
EVENT_INVESTOR
EventEngine
Put()
Put()
Put()
__onTimer()
__queue
Event
Type
Dict
Event
Type
Dict
Event
Get()
__run()
__process
Type
handler1
handler2
handler3
Dict
__init()
Start()
__active
__thread
Stop()
__handlers
Key
Value
Type_
handlerList
handler1
handler2
handler3
Type_
handlerList
handler
handler
handler
Type_
handlerList
handler
handler
handler



备注：

- 黑色是启动的过程。
- 紫色是tick数据的流。
- 绿色是下单，买开，买平，卖平，卖开.（该功能缺失，在TdApi中需要订阅订单更新事件，并驱动UpdateOrder()
- 蓝色是成交单。（该功能缺失，在TdApi中需要订阅成交事件，并驱动UpdateTrade()
- 红色是止损单。



备注：

- 黑色是启动的过程，或调用过程。
- 紫色是tick数据的流。
- 绿色是下单，买开，买平，卖平，卖开
- 蓝色是成交单
- 红色是止损单。
- 橙色是过程数据

ContractMonitor: BasicMonitor
合约查询组件

dataEngine	#数据引擎
__init()__	
initUi()	#初始化界面
showAllContract()	#显示所有合约数据
reflash()	#刷新
initMenu()	#初始化右键菜单
contextMenuEvent()	#右键鼠标点击
Show()	#显示

TradeingWidget: QFrame
简单交易组件

mainEngine	#主引擎
eventEngine	#事件引擎
dataEngine	#数据引擎
Symbol	#合约代码
__init()__	
initUi()	#初始化界面
updateSymbol()	#合约更新
updateTick()	#行情更新
connectSignal	#连接signal
sendOrder()	#发单
cancelAll()	#一键取消所有委托

MarketMonitor: BasicMonitor
市场监控组件

__init()__	

TradeMonitor: BasicMonitor
交易监控组件

__init()__	

OrderMonitor: BasicMonitor
委托监控组件

__init()__	

LogMonitor: BasicMonitor
日志监控组件

__init()__	

BasicMonitor:QTableWidgetItem
基础监控

eventEngine	#事件引擎
headerDict	#表头标签字典
headerList	#表头标题列表
eventType	#监控事件类型
font	#字体
saveData	#是否保存数据到单元格
setHeaderDict()	#设置表头字典
setDataKey()	#设置数据字典Key
setEventType()	#设置事件类型
setFont()	#设置字体
setSaveData()	#设置是否保存数据
initTable ()	#初始化表格
registerEvent()	#注册GUI更新事件
updateEvent()	#收到事件更新
updateData()	#更新数据
resizeColumn()	#调整各列的大小

MainWindows : QtMainWindow
主窗口

mainEngine	#主引擎
eventEngine	#事件引擎
dataEngine	#数据引擎
initUi()	#初始化界面
initCentral()	#初始化中心区域
initMenu ()	#初始化菜单
initStatusBar()	#初始化状态栏
updateStatusBar()	#更新CPU/内存
getCpuMemoyt()	#查询Cpu/内存
connectCtp()	#连接CTP接口
connectLts()	#连接LTS接口
connectWind()	#连接Wind接口
openAbout()	#打开关于
openContract()	#打开合约查询
closeEvent()	#关闭事件

PositionMonitor: BasicMonitor
仓位监控组件

__init()__	

AccountMonitor: BasicMonitor
账户监控组件

__init()__	

ErrorMonitor: BasicMonitor
错误监控组件

__init()__	

DirectionCell:QTableWidgetItem
显示买卖方向的单元格

data	#数据
text	#文本内容
setContent()	#设置文本内容

BasicCell:QTableWidgetItem
基础单元格

data	#数据
text	#文本内容
setContent()	#设置文本内容

NameCell:QTableWidgetItem
显示合约中文的单元格

data	#数据
text	#文本内容
setContent()	#设置文本内容

BidCell:QTableWidgetItem
显示买价的单元格

data	#数据
text	#文本内容
setContent()	#设置文本内容

AskCell:QTableWidgetItem
显示卖价的单元格

data	#数据
text	#文本内容
setContent()	#设置文本内容

ValueMonitor: QTableWidget
数值监控组件

__init()__	

CtaStrategyManager: QGroupBox
策略管理组件

__init()__	

CtaEngineManager: QWidget
策略引擎管理组件

__init()__	

StopOrder(ctaEngine.Py)
本地停止单

vtSymbol	#vt系统代码
orderType	#类型
price	#价格
volume	#数量
strategy	#下单策略对象
stopOrderId	#本地编号
status	#状态

Event 事件类

type	#事件类型
dict	#事件数据字典

CtaBarData(ctaEngine.Py)
K线数据

vtSymbol	#vt系统代码
symbol	#代码
exchange	#交易所
open	#开市价
high	#最高价
low	#最低价
close	#收市价
date	#bar开始日期
time	#时间
datetime	#日期时间
volume	#成交量
openInterest	#持仓量

CtaTickData(ctaEngine.Py)
Tick数据

vtSymbol	#vt系统代码
symbol	#代码
exchange	#交易所
lastPrice	#最新价
volume	#最新成交量
openInterest	#持仓量
upperLimit	#涨停价
lowerLimit	#跌停价
date	#bar开始日期
time	#时间
datetime	#日期时间
bidPrice1	#卖价1
bidPrice2	#卖价2
bidPrice3	#卖价3
bidPrice4	#卖价4
bidPrice5	#卖价5
askPrice1	#买价1
askPrice2	#买价2
askPrice3	#买价3
askPrice4	#买价4
askPrice5	#买价5
bidVolume1	#卖出数量1
bidVolume2	#卖出数量2
bidVolume3	#卖出数量3
bidVolume4	#卖出数量4
bidVolume5	#卖出数量5
askVolume1	#买入数量1
askVolume2	#买入数量2
askVolume3	#买入数量3
askVolume4	#买入数量4
askVolume5	#买入数量5

CtaEngine(ctaEngine.Py)
CTA策略引擎

mainengine	#主引擎（引用）
eventengine	#事件引擎（引用）
dataengine	#数据引擎（引用）
strategyDict	#策略对象字典
tickStrategyDict	#Tick推送策略字典
orderStrategyDict	#下单与策略映射字典
stopOrderCount	#本地停止单计数器
stopOrderDict	#本地停止单字典(持久)
workingStopOrderDict	#本地停止单字典
sendOrder()	#发单
cancelOrder()	#撤单
sendStopOrder()	#发本地停止单
cancelStopOrder()	#撤销本地停止单
processStopOrder()	#实时处理停止单
processTick()	#处理行情推送
processOrder()	#处理委托推送
processTrade()	#处理成交推送
registerEvent()	#注册事件监听
insertData()	#插入数据到数据库
loadBar()	#从数据库读取Bar数据
loadTick()	#从数据库读取Tick数据
getToday()	#获取当前日
writeCtaLog()	#写入日志事件
initStrategy()	#初始化策略
startStrategy()	#启动策略
stoptStrategy()	#停止策略
saveStrategySetting()	#保存策略配置
loadStrategySetting()	#读取引擎中的策略配置
getStrategyVarialbe()	#获取策略当前变量字典
getStrategyParameter()	#获取策略的参数字典

CtaTemplate CTA策略模板（基类）

strategyClassName	#策略类名称
parmList	#参数列表
varList	#变量列表
vtSymbol	#策略的合约vt系统代码
ctaEngine	#策略引擎对象
tickDbName	#Tick数据库名称
barDbName	#Bar数据库名称
trading	#策略是否启动交易
onTick()	#行情更新事件
onTrade()	#交易更新事件
onOrder()	#报单更新事件
onBar()	#K线数据更新
onTimer()	#定时器事件
onInit()	#初始化策略
onStart()	#启动交易
onStop()	#停止交易
buy()	#买入开仓
cover()	#买入平仓
sell()	#卖出平仓
short()	#卖出开仓
cancelOrder()	#撤单
insertTick()	#向数据库中插入Tick数据
insertBar()	#想数据库中插入Bar数据
loadTick()	#读取Tick数据
loadBar()	#读取Bar数据
setParam()	#设置参数
getToday()	#查询当前日期

EventEngine
事件引擎

__queue	#事件队列
__active	#事件引擎开关
__thread	#事件处理线程
__timer	#计时器
__handlers	#事件、调用函数 映射字典
__init	#初始化事件
__run	#引擎运行
__process	#处理事件
__onTimer	#向事件队列存入计数器事件
start	#引擎启动
stop	#停止引擎
register	#注册事件处理函数监听
unregister	#注销事件处理函数监听
put	#事件队列

VtGateway
交易接口(基类)

eventEngine	#事件引擎
onTick()	#市场推送行情
onTrade()	#成交信息推送
onOrder()	#订单变化推送
onPosition()	#持仓信息推送
onAccount()	#账户信息推送
onError()	#错误信息推送
onLog()	#日志推送
onContract()	#合约基础信息推送
connect()	#连接
subscribe()	#订阅行情
sendOrder()	#发单
cancelOrder()	#撤单
qryAccount()	#查询账户资金
qryPosition()	#查询持仓
close()	#关闭

CtpGateway : vtGateway
CTP接口

eventEngine	#事件引擎
mdApi	#行情Api
tdApi	#交易Api
mdConnected	#行情Api连接状态
tdConnected	#交易Api连接状态
onTick()	#市场推送行情
onTrade()	#成交信息推送
onOrder()	#订单变化推送
onPosition()	#持仓信息推送
onAccount()	#账户信息推送
onError()	#错误信息推送
onLog()	#日志推送
onContract()	#合约基础信息推送
connect()	#连接
subscribe()	#订阅行情
sendOrder()	#发单
cancelOrder()	#撤单
qryAccount()	#查询账户资金
qryPosition()	#查询持仓
close()	#关闭
initQuery()	#初始化连续查询
query()	#注册在事件引擎的查询函数
startQuery()	#启动连续查询
setQryEnabled()	#设置启动循环查询

CtpMdApi

CtpTdApi

windGateway : vtGateway
Wind接口

eventEngine	#事件引擎
w	#w
Connected	#连接状态
wsqParamMap	#参数传入字段列表
tickDict	#本地Tick快照
connect()	#连接
subscribe()	#订阅行情
sendOrder()	#发单
cancelOrder()	#撤单
getAccount()	#查询账户资金
getPosition()	#查询持仓
close()	#关闭
wConnect()	#异步连接
wsqCallBack()	#Wsq推送

LtsGateway : vtGateway
Lts接口

eventEngine	#事件引擎
mdApi	#行情Api
tdApi	#交易Api
qryApi	#查询Api
mdConnected	#行情Api连接状态
tdConnected	#交易Api连接状态
qryConnected	#查询API连接状态
onTick()	#市场推送行情
onTrade()	#成交信息推送
onOrder()	#订单变化推送
onPosition()	#持仓信息推送
onAccount()	#账户信息推送
onError()	#错误信息推送
onLog()	#日志推送
onContract()	#合约基础信息推送
connect()	#连接
subscribe()	#订阅行情
sendOrder()	#发单
cancelOrder()	#撤单
qryAccount()	#查询账户资金
qryPosition()	#查询持仓
close()	#关闭
initQuery()	#初始化连续查询
query()	#注册在事件引擎的查询函数
startQuery()	#启动连续查询
setQryEnabled()	#设置启动循环查询

LtsQryApi

LtsMdApi

LtsTdApi

MainEngine
主引擎

eventEngine	#事件引擎
dataEngine	#数据引擎
gatwayDict	#接口对象字典
dbClient	#数据库客户端对象
addGateway()	#添加接口
connect()	#连接特定名称的接口
subscribe ()	#订阅合约
sendOrder()	#发单（委托下单）
cancelOrder()	#撤单
getAccount()	#查询账户
getPosition()	#查询持仓
exit()	#退出程序时调用
writeLog()	#发出日志事件
dbConnect()	#链接数据库
dbInsert()	#向数据库插入数据
dbQuery	#保从数据库取数据

DataEngine
数据引擎

eventEngine	#事件引擎
cotractDict	#合约详细字典
orderDict	#委托数据的字典
workingOrderDict	#活动委托数据的字典
updateContract()	#更新合约数据
getContract()	#查询合约数据
getAllContract()	#查询所有合约数据
saveContracts()	#保存合约到本地硬盘
loadContracts()	#从硬盘读取合约
updateOrder()	#更新委托数据
getOrder()	#查询委托数据
getAllWorkingOrder()	#查询所有活动委托数据
registerEvent	#注册事件监听

vtErrorData: vtBaseData
错误数据类

errorID	#错误代码
errorMsg	#错误信息
additionalInfo	#补充信息

vtLogData: vtBaseData
日志数据类

logTime	#日志生成时间
logContent	#日志信息

vtContractData: vtBaseData
合约详细信息类

symbol	#合约代码
exchange	#交易所代码
vtSymbol	#合约在VT系统的代码
name	#合约中文名
productClass	#合约类型
size	#合约大小
priceTick	#合约最小价格tick
strikePrice	#期权行权价
underlyingSymbol	#表弟物合约代码
optionType	#期权类型

vtOrderData: vtBaseData
订单数据类

symbol	#合约代码
exchange	#交易所代码
vtSymbol	#合约在VT系统的代码
orderID	#订单编号
vtOrdeddID	#订单在vt系统的编号
direction	#成交方向
offset	#成交开平仓
price	#成交价格
totalVolume	#报单总数量
tradeVolume	#报单成交数量
status	#报单状态
orderTime	#发单时间
cancelTime	#撤单时间
frontID	#前置机编号
sessionID	#连接编号

vtTradeData: vtBaseData
成交数据类

symbol	#合约代码
exchange	#交易所代码
vtSymbol	#合约在VT系统的代码
tradeID	#成交编号
vtTradeID	#成交在vt系统的编号
orderID	#订单编号
vtOrdeddID	#订单在vt系统的编号
direction	#成交方向
offset	#成交开平仓
price	#成交价格
volume	#成交数量
tradeTime	#成交时间

vtBaseData
回调函数推送数据的基础类

gatewayName	#接口名称
rawData	#原始数据

vtPositionData: vtBaseData
持仓数据类

symbol	#合约代码
exchange	#交易所代码
vtSymbol	#合约在VT系统的代码
direction	#持仓方向
position	#持仓数量
frozen	#冻结数量
price	#持仓均价
vtPositionName	#持仓在vt系统中的代码

vtAccountData: vtBaseData
账户数据类

accountID	#账户代码
vtAccountDl	#账户在VT系统的代码
preBalance	#昨日账户结算净值
balance	#账户净值
available	#可用资金
commition	#今日手续费
margin	#保证金占用
closeProfit	#平仓盈利
positionProfiit	#持仓盈利

vtSubScribeReq 订阅行情对象类

symbol	#代码
exchange	#交易所

VtTickData : vtBaseData
Tick行情数据类

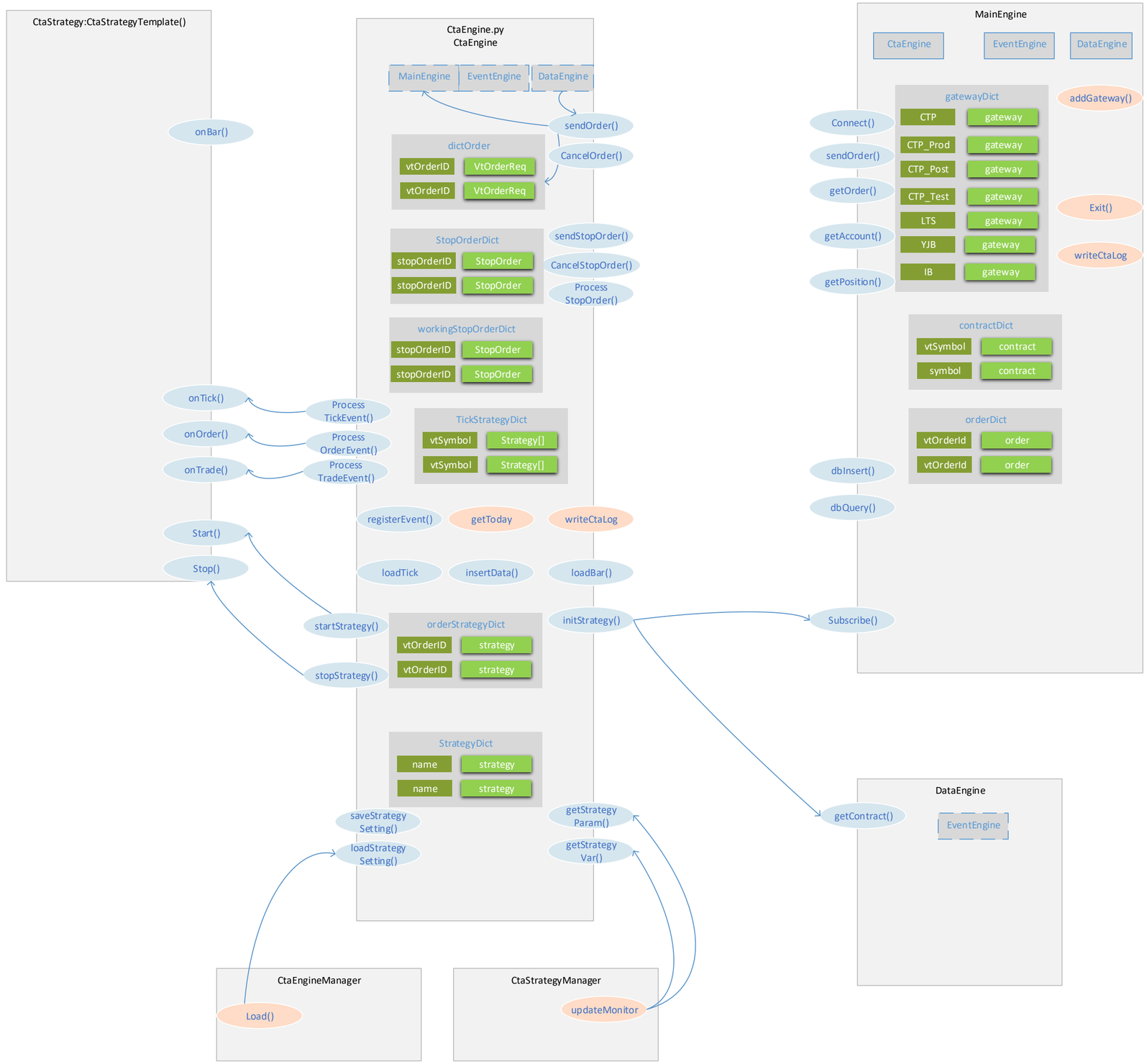
vtSymbol	#vt系统代码
symbol	#代码
exchange	#交易所
lastPrice	#最新价
volume	#最新成交量
openInterest	#持仓量
upperLimit	#涨停价
lowerLimit	#跌停价
date	#bar开始日期
time	#时间
datetime	#日期时间
bidPrice1	#卖价1
bidPrice2	#卖价2
bidPrice3	#卖价3
bidPrice4	#卖价4
bidPrice5	#卖价5
askPrice1	#买价1
askPrice2	#买价2
askPrice3	#买价3
askPrice4	#买价4
askPrice5	#买价5
bidVolume1	#卖出数量1
bidVolume2	#卖出数量2
bidVolume3	#卖出数量3
bidVolume4	#卖出数量4
bidVolume5	#卖出数量5
askVolume1	#买入数量1
askVolume2	#买入数量2
askVolume3	#买入数量3
askVolume4	#买入数量4
askVolume5	#买入数量5

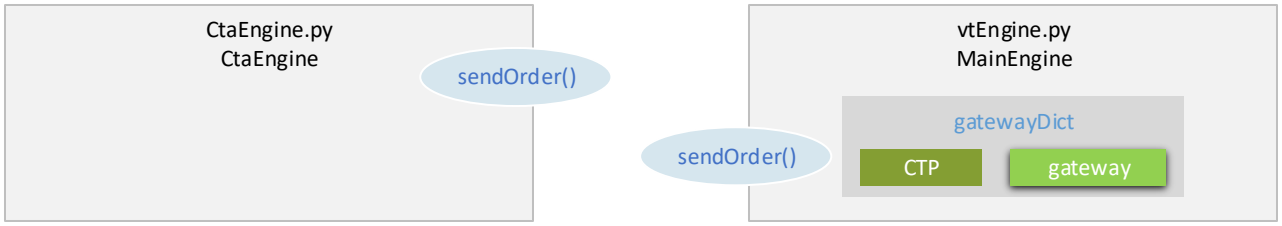
vtOrderReq 发单时传入对象类

symbol	#代码
exchange	#交易所
price	#价格
volume	#数量
priceType	#价格类型
direction	#买卖方向
offset	#开平

vtCancelOrderReq 撤单时传入对象类

symbol	#代码
exchange	#交易所
OrderID	#报单号
frontID	#前置机
sessionID	#会话号


CtaStrategy:CtaStrategyTemplate()
onBar()
onTick()
onOrder()
onTrade()
Start()
Stop()
CtaEngine.py
CtaEngine
MainEngine
EventEngine
DataEngine
sendOrder()
CancelOrder()
dictOrder
vtOrderID
VtOrderReq
StopOrderDict
stopOrderID
StopOrder
sendStopOrder()
CancelStopOrder()
Process StopOrder()
workingStopOrderDict
Process TickEvent()
Process OrderEvent()
Process TradeEvent()
TickStrategyDict
vtSymbol
Strategy[]
registerEvent()
getToday
writeCtaLog
loadTick
insertData()
loadBar()
startStrategy()
stopStrategy()
orderStrategyDict
strategy
initStrategy()
StrategyDict
name
saveStrategy Setting()
loadStrategy Setting()
getStrategy Param()
getStrategy Var()
MainEngine
CtaEngine
EventEngine
DataEngine
gatewayDict
CTP
gateway
CTP_Prod
CTP_Post
CTP_Test
LTS
YJB
IB
addGateway()
Connect()
sendOrder()
getOrder()
getAccount()
getPosition()
Exit()
writeCtaLog
contractDict
vtSymbol
contract
symbol
orderDict
vtOrderId
order
dbInsert()
dbQuery()
Subscribe()
DataEngine
getContract()
EventEngine
CtaEngineManager
Load()
CtaStrategyManager
updateMonitor


CtaEngine.py
CtaEngine
sendOrder()
vtEngine.py
MainEngine
gatewayDict
sendOrder()
CTP
gateway