**Strategy:**

根据输入的市场数据,发出买卖信号的逻辑实体.

**Parameter:**

策略依据参数值调整信号.

**Model:**

一个策略配合一组参数构成一个模型

**Instrument:**

一个可交易的具体合约,如IF1502

**Symbol<deprecated>:**

一个交易品种,如果IF,au等,它包含多个子Instrument.如IF下有IF1412,IF1502等.

**Contract:**

关于交易品种的信息，如手续费、最小跳、乘数、交易时间段等。

**BarFeed:**

一个Instrument的Bar时间序列。

**DataStream:**

一个品种下所有同样Resolution和Interval的数据集合。

可能包含一个或多个BarFeed的数据序列.一个DataStream中的所有BarFeed具有相同的Resolution.

DataStream的作用之一是将多个BarFeed打包成一组，这样可以灵活地进行各种数据组合。

一个DataStream中的数据应具有相同的类型和周期，例如，属于同一品种、时间周期一样、手续费一样、最小跳一样等。

一个DataStream绑定一个Contract。

为了方便使用，一个DataStream可以只包含一个连续主力合约，所有不同月份的合约组成一个时间连续、命名统一的BarFeed。

一个Strategy至少要订阅一个DataStream，否则它将不会有数据输入。

**TS File:**

可以映射到内存的二进制文件,一个TS(或CSV)文件仅包含一个DataStream.

**Bin File:**

可以直接映射到内存的不带头信息的二进制文件，该文件存储Bar序列数据。

一个Bin File可以存储多个Instrument序列数据，它们在文件中连续存储，但在时间上允许有重叠。同一个Instrument内的Bar序列按时间顺序连续存放在一起。每个Instrument的Bar数据中有一个字段指示当前Bar是否是可交易的（可简单地理解为是否是主力合约，因为我们只交易主力合约，所以在某个时间点上只有主力合约是可交易的），程序在扫描Instrument序列生成BarFeed时，需要读取该字段，最终形成一个或多个BarFeed可交易时间段信息，并保存在BarFeed中。Runtime在订阅BarFeed时会读取可交易时间段信息，在可交易时间段之外的报单将直接被忽略。

由于主力合约经常来回切换，所以一个Instrument的可交易时间段可能会由两个到多个不连续时间段组成。

一个Bin File经常用于存储一个品种的所有主力合约。这些主力合约具有相同的Resolution，例如一分钟。例子参见文件MainInstruments.txt，该文件指明了Bin File中合约数据是如何排列的。

**BarStorage:**

一个BarStorage可以加载多个TS File,也就是说,一个BarStorage可以包含多个DataStream,一个DataStream可包含多个BarFeed,一个BarFeed包含唯一一个Instrument.

**Runtime:**

一个Runtime包含唯一个Strategy,是Strategy实例的运行环境.

**Process:**

一个DataStream与一个Model,组成一个Process.

一个Process是一个Model在一个(主力)DataStream上的一次运行.由于一个DataStream可能包含多个BarFeed,从而可能包含多个Instrument,所以一个Process可能生成多个在不同Instrument上交易的Runtime.

**Executor:**

执行器.执行器包含一个Broker和多个BarFeed,并包含至少一个Process.

执行器负责将品种,模型和参数打包和分配到各个Process中,并将它们组合起来进行测试.

一个Executor代表了一个组合测试.

Executor按时间先后顺序分发各个BarFeed中的Bar数据.

**Backtester:**

回测器.简单地执行一个Executor.

**Optimizer:**

优化器.分解品种,模型和参数,生成众多Executor,并执行它们,根据Executor的绩效报告找出最优参数组合.

**Position:**

策略的仓位。

该抽象只在Runtime中可见。

xBacktest的设计大致可分为两层，模拟现实中交易所与柜台分离的架构。交易所与柜台都会进行核算。

**SubPosition:**

子仓位。

构成Position的子部件。每一个成交的订单都会生成一个SubPosition，每个Position至少包含一个SubPosition。之所以抽象出子仓位，是为了支持策略的加、减仓操作。

Strategy支持平掉某个指定的子仓位，但这在现实中是不可行的。在实际交易中，仓位根据先开先平的原则进行操作，平仓时无法指定平掉仓位的哪个部分。

xBacktest在回测中支持平掉仓位的指定部分(SubPosition)，虽然这与现实情况不符，但作为一种方便的抽象，它是有用的。并且这样的规则也不会导致账户权益与现实不符。