Price Publication Service

Sequence diagram

cleanUp

prices

getPricesByInstrument

prices

getPricesByVendor

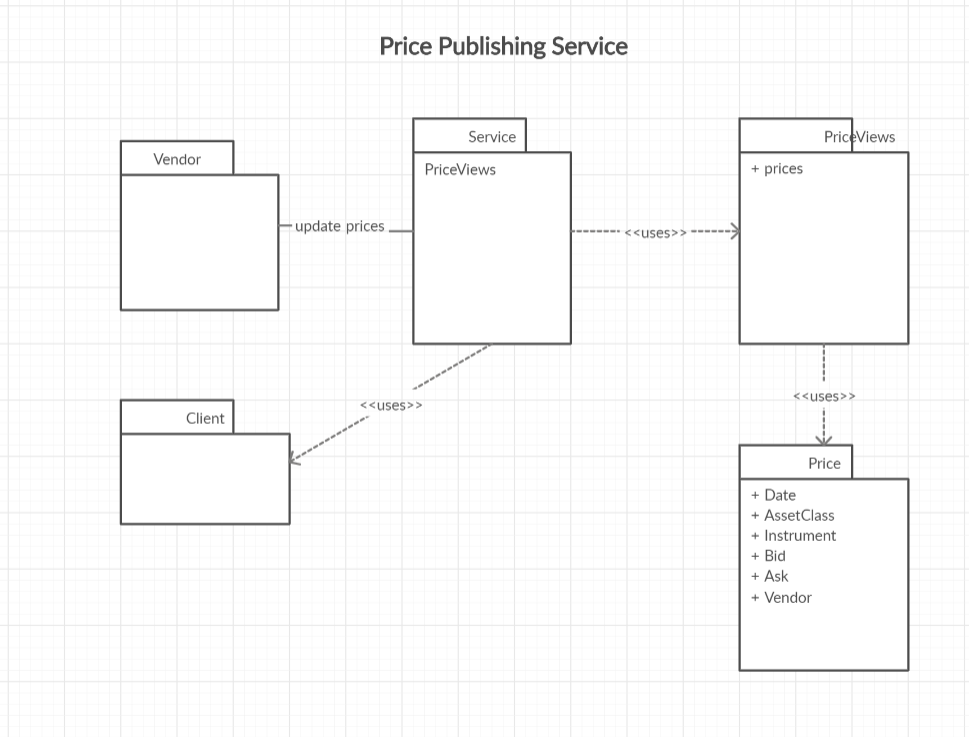
Vendor

Update

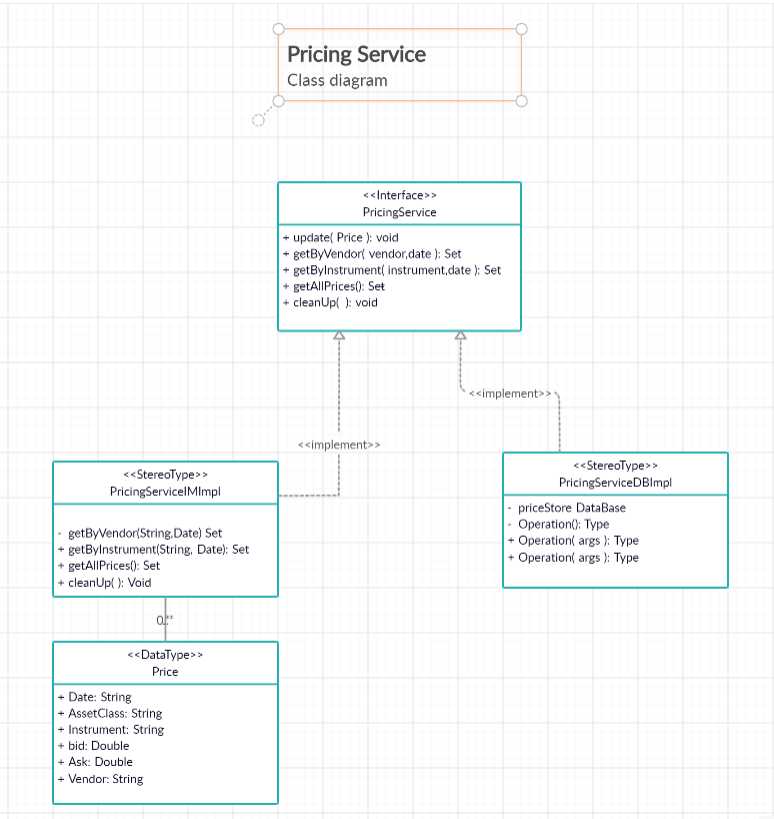
PricingService

EODService

Client



Domain model



# Developer Note:

The PricingService can be exposed as a REST service by wiring it in SpringBoot Application. For the purpose of this exercise, the service can be invoked by the Junit tests.

# Development requirement and Analysis log

Satkuru:

1.       The frequency of the price update from the vendors

2.       The requirement states, "The cache will have services to allow clients to publish and retrieve data from the store", does this mean the service must allow clients to publish price to the service as well (in addition to various vendors, kindly clarify).

3.       Do the prices need to be conflated? (i.e. clients always get the latest data and we discard older data)

4.       Is the price specific to a date (As of Date)?

5.       what details are provided as part of vendor price update (market, asset class, etc.). this is to model the price’s cacheable object.

6.       How expired (after 30 days) prices should be evicted from the cache. Does it need to be automatic (with some demon thread running the background) or can we run some EOD process that remove these expired instruments? To keep this exercise simple, I’d recommend the latter.

7.       Number of instruments and vendors - how much do you need to scale?

8.       Latency. It will affect design - ultra low latency will need "local caching" of the prices plus non-blocking data structures if not low latency, cache can be scaled using micro-services if required.

9.       The latency will also impact the data structures we use to store data - since clients can ask for "all data per vendor" or "one instrument across vendors", we could actually use 2 different maps to store the 2 different views.

10.   Is this going to be only one price for simplicity or will have the usual buy/sell prices?

11.   For more advanced use cases - would clients ask for "best" price per instrument - so sort in some order?

12.   Does the cache service need to backed by some permeant persistence in the background, which will allow service to be brought down and then brought up?

Jonathan:

1.      In this exercise, you can assume that there would usually be one price update per instrument and vendor per day. **[SK]**Does it mean when  client requests prices  (either by vendor or by instrument ), the client need s provide a date or does the service always provides the latest prices available ?[JG] The client should have the option of specifying the date they require otherwise you should return the latest.

2.      You can consider publishers and consumers of prices to be distinct types of clients.

3.      Yes.

4.      Yes.

5.      We can keep it simple for this exercise. Beyond the basic identifying information for the instrument and vendor, you can include the market and asset class fields (although you can keep the implementation for the exercise simple and just support equities) and price type. **[SK]**Does a single price contain both bid and ask prices together or are they updated separately. [JG] It’s acceptable to either have the price type as a parameter in the request or to ensure if both are returned in all cases, they are clearly identified.[SK] Just to clarify my question regarding point #5, I was referring to the price updates receive from vendors. However, it is safe to assume the price received will contain both bid and ask prices as per market data convention.  Kindly correct me if it is otherwise. [JG] Yes, you will receive both bid and ask prices from the vendors.

6.      This is up to you.

7.      Initially you can assume ~1000 instruments with 5 vendors with the expectation that the number of instruments could reach the low 10s of thousands in the not too distant future.

8.      Do not attempt to design for ultra-low latency in your solution as it’s unlikely the time constraints of the exercise will allow for this. It is worth considering how you might later evolve the design to deal with much more demanding latency requirements in future, however.

9.      This reads more like a statement than a question.  [SK] sorry it was meant to be an observation based on the design approach (related to point #8).

10.   You should differentiate bid and ask prices.

11.   Not in this exercise. Good question, though.

12.   Not initially, but please note the requirement in the problem definition for the backing store to be easily substitutable for something else.