Coinbase Take Home Exercise

Instructions

Your goal is to create a web service that provides quotes for digital currency trades using data from the GDAX orderbook.

GDAX, our digital currency exchange, maintains an order book for each tradable currency pair (eg. BTC-USD). An order book is comprised of a series of bids (offers to buy) and asks (offers to sell). Bids are sorted descending by price (highest price first) and asks are sorted ascending by price (lowest price first). If these two numbers ever cross, a trade is executed and those bids or asks are removed from the order book. The GDAX API exposes an endpoint to retrieve the current order book for each currency pair. For this task, you should use the level 2 query parameter to fetch aggregated order information. *Note: this API can be accessed unauthenticated; you do not need API keys or a GDAX account to access it.*

Your service will handle requests to buy or sell a particular amount of a currency (the base currency) with another currency (the quote currency). The service should use the orderbook to determine the best price the user would be able to get for that request by executing trades on GDAX. Note that the quantity your user enters will rarely match a quantity in the order book exactly. This means your code will need to aggregate orders in the order book or use parts of orders to arrive at the exact quantity, and your final quote will be a weighted average of those prices.

Service Specification

The web service has only one endpoint that receives JSON requests and responds with JSON. If there are any errors processing the request, it responds with a JSON object including an error message.

Route	POST /quote
Request fields	• action (String): Either "buy" or "sell" • base_currency (String): The currency to be bought or sold • quote_currency (String): The currency to quote the price in • amount (String): The amount of the base currency to be traded
Response fields	• price (String): The per-unit cost of the base currency • total (String): Total quantity of quote currency • currency (String): The quote currency

The service should be able to quote trades between any two currencies that GDAX has an orderbook for. It should also be able to support trades where the base and quote currencies are the inverse of a GDAX trading pair. For example, the service should be able to quote a buy of BTC (base currency) using ETH (quote currency), even though the GDAX orderbook is ETH-BTC.

Examples

Request	Response

{
"action": "buy",
"base_currency": "BTC",
"quote_currency": "USD",
"amount": "1.00000000"
}
	{
	"total": "705.40",
	"price": "705.40",
	"currency": "USD"
	}

{
"action": "sell",
"base_currency": "BTC",
"quote_currency": "USD",
"amount":
"10.00000000"
}
	{
	"total": "7052.03",
	"price": "705.20",
	"currency": "USD"
	}

{
"action": "buy",
"base_currency": "USD",
"quote_currency": "BTC",
"amount": "1000.00"
}
	{
	"total": "1.41783638",
	"price":
	"0.00147836",
	"currency": "BTC"
	}

In the previous example, the trade would be executed against the BTC-USD order book despite the base currency and quote currency being reversed.

Submission

You can use any language and framework of your choice, we recommend you choosing the ones you are most proficient in. The only requirement is that it can be set up and run on a Unix environment.

When you are done, email back a Dropbox or similar link to your source code and instructions to run the project in a Unix environment as well as how much time you spent on the problem. If there are any issues during setup, we may reach out to you.

Evaluation

Your submission will be evaluated on:

- Adherence to specifications
- Code quality, documentation and readability
- Idiomatic usage of the language or framework of your choice
- Data structures, design patterns, and architectural tradeoffs
- Handling of edge cases and errors