Algo Trade Signal Handler

Requirements

1. A service that processes algorithm trading signals over HTTP, where a trading signal is an integer passed to the service.
2. Upon receiving a trading signal, the service uses a predefined configuration to trigger the execution of the Algo, where the Algo is provided by an external library with a predefined interface.
3. For each provided signal a new instance of the Algo library must be instantiated.
4. If the signal provided is recognised and so has a predefined Algo configuration, it will require the execution of the following methods against the external Algo library, in a defined order:
    1. algo.doAlgo()
    2. algo.reverse()
    3. algo.submitToMarket()
    4. algo.performCalc()
    5. algo.setUp()
    6. algo.setAlgoParam(int param, int value)
        1. If algo.setAlgoParam is required as part of the configuration, then a set of param-value pairs will be provided in the configuration.
        2. The algo.setAlgoParam method must be called with each of the param-value pairings in the configuration.
5. If the signal provided is not recognised and so does not have a predefined Algo configuration, then algo.cancelTrades() must be invoked.
6. After the above actions are taken algo.doAlgo() must be invoked
7. It must be easy and non-disruptive to define Algo Configurations to use for new signals. These algo configurations are currently provided by analysts via JIRA tickets.
8. The service must be easy to understand and debug.
9. The service must be easy to test and verify.
10. The service must be able to handle 50 new signal configurations per month (600 after year one, 1200 after year two etc).

Assumptions

1. The service is an internal service and does not need additional authentication. Access to the service will be limited to authorised callers.
2. A signal may trigger any number of the calls defined in requirement 4, in any order.

Solution Design

Each signal id (integer) will correspond to an Algorithm configuration. The configuration will store all actions to be taken upon receiving the signal assigned against the configuration.

An algorithm configuration will be defined as an ordered list of steps to execute against the Algo library before calling doAlgo.

Each of the steps defined against an algorithm configuration corresponds to one of the method calls made against the Algo class. Below is a map from the step name to the method to be invoked against the Algo class:

- SET_UP -> algo.setUp()
- REVERSE -> algo.reverse()
- PERFORM_CALC -> algo.performCalc()
- SUBMIT_TO_MARKET -> algo.submitToMarket()
- SET_ALGO_PARAM -> algo.setAlgoParam(int param, int value)
  - This step requires param-value pairings as arguments.
  - If this step is required as part of the algo configuration then the algo configuration will contain a set of these param-value pairings. When the step is required then the algo.setAlgoParam method will be invoked once per param-value pairing, with the specified param & value arguments.

Algorithm configurations will be stored in a mySql database. This allows them to be easily added without having to modify and redeploy the service code.

The database will contain 3 tables

- Algo_config
  - stores the signal_id integer as the pk.
- Algo_config_param
  - stores the param_id and param_value integer pairs
  - references Algo_config via a signal_id foreign key.
  - Pk across the signal_id and the param_id.
- Algo_config_step
  - Stores a step_type from the above list and a sequence_index to define the order in which the steps should be executed
  - references Algo_config via a signal_id foreign key.
  - Pk across signal_id and sequence_index.

Algorithm configurations will be cached in-memory such that we do not need to request the configuration on each signal.

The algorithm configuration cache will be filled upon service launch, such that at launch time all currently registered configurations will be loaded. Any configurations added to the database after the service has started will be loaded when a signal requiring that configuration is received.

The service will be a Spring Boot service with one POST endpoint '/signal/{signalId}.

When the endpoint is invoked:

- A new instance of the Algo class is created.
- the service will retrieve the configuration for that signalId. It will first check the configuration cache, and then it will attempt to load the configuration from the MySQL database.
- If a configuration for the signalId integer is found then the steps contained in the configuration will then be invoked against the Algo class.
- If a configuration is not found then cancelTrades() is invoked against the Algo.
- Finally doAlgo is invoked against the Algo
- 200 status is returned.

Spring Data JPA and Hibernate will be used as an ORM to access the configurations from the MySQL database.

Possible improvements

1. Integration tests - test the service end to end.
2. Validation to ensure that the configurations loaded only contain each step once.
3. Error handling - custom Exceptions thrown and then handled with a controller advice exception handler.
4. Extract the AlgoConfigurationCache out of the AlgoConfigurationService
5. An endpoint to add new AlgoConfigurations, or perhaps a separate service will manage that.
6. More debug / info logging.

Questions

1. Is the submitToMarket() call always made regardless of the rest of the configuration? If so that would be removed from the AlgoStepType enum and AlgoConfiguration and the submitToMarket call would be invoked by the SignalHandlerService after using the AlgoConfiguration to configure the Algo.
2. Do the setAlgoParam calls need to be made in a specific order?