Database Access Layer and Persistence

- Product
- Market Data Nexus
- CEP
- Smart Order Routing
- Strategy Engines
- DARE
- Photon
- System Administration
- Database Access Layer and Persistence
- Threading Model
- High Availability and Failover

DATABASE ACCESS LAYER AND PERSISTENCE

Database Access Layer and Persistence

The Marketcetera Automated Trading Platform uses Hibernate Object Relational Mapping to provide access to the database. Hibernate ORM (Hibernate in short) is an object-relational mapping framework for the Java language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions (source). In addition to Hibernate, the MATP uses Spring Data JPA to more easily construct and execute database queries.

Database access is organized into layers. A Service class provides transactions and logical grouping of database queries to accomplish a business goal. A DAO interface (Data Access Object) provides the actual database query, using Spring Data JPA. Transactions are indicated using AOP annotations when possible, otherwise, they are manually constructed and committed, when annotations are not possible to use. Two scenarios where it's not possible to use AOP annotations are: first, if the transaction you want to execute is in a non-public method or class; second, if the Service method needs finer granularity than the method itself for the transaction. An example of the first scenario is the process used to check aggregations to see if they should be terminated. The method in AggregationServiceImpl that is run is run by a timer, which is not visible to AOP. Therefore, a manual transaction is needed. An example of the second scenario is BrokerServiceImpl.disableSession which disables a FIX session. There, two transactions are needed, making it a poor match for AOP annotations. The first transaction updates the database with the new status, the second is used to cover notifying the other cluster members of the disabled session.

Not every service has a corresponding web service implementation. Most services don't need one. Any service that is provided to the UI or outside the platform does have a web service implementation. The web service implementation never handles transactions or database access. The service implementation never handles web services or database access. Each layer has a specific purpose.

Any ACID-compliant database with a modern JDBC driver can be used. The current implementation uses Oracle via the Oracle-provided thin client JDBC driver. Database connections are managed by a connection pool provided by the C3P0 library. Using a database connection pool prevents the application from being required to create a new database connection for every call, which is expensive.

Complicated queries and joins that cannot be managed effectively with Spring Data JPA are managed by an additional library called QueryDSL. QueryDSL adds some generated classes at build-time that allow complex queries to be built and executed with code. For example, some of the more complicated queries that drive the Trader Blotter display are performed using QueryDSL.