Other languages: 繁體中文

Database - General

General

Drogon has built-in database read/write engine. The operation of database connection is based on non-blocking I/O technology. Therefore, the application works in an efficient non-blocking asynchronous mode from the bottom to the upper layer, which ensures Drogon's high concurrency performance. Currently, Drogon supports PostgreSQL and MySQL databases. If you want to use a database, the development environment of the corresponding database must be installed first. Drogon will automatically detect the header files and library files of these libraries and compile the corresponding parts. For the preparation of the database development environment, see Development Environment.

Drogon supports the sqlite3 database in order to support lightweight applications. The asynchronous interface is implemented through the thread pool, which is the same as the interfaces of the aforementioned databases.

DbClient

The basic class of Drogon's database is `DbClient` (this is an abstract class, the specific type depends on the interface that constructs it). Unlike a generic database interface, a `DbClient` object does not represent a single database connection. It can contain one or more database connections, so you can think of it as a **connection pool object**.

`DbClient` provides both synchronous and asynchronous interfaces. The asynchronous interface also supports both blocking and non-blocking modes. Of course, for the cooperation with the Drogon asynchronous framework, it is recommended that you use the the asynchronous interface with non-blocking mode.

Usually, when an asynchronous interface is called, `DbClient` will randomly select one of the idle connections it manages to perform related query operations. When the result returns, `DbClient` will process the data and return it to the caller through the callback function object; Without an idle connection, the execution content will be cached. Once a connection has executed its own sql request, the pending command will be fetched from the cache to execute.

For details on `DbClient`, see DbClient.

Transaction

The transaction object can be generated by `DbClient` to support transaction operations. In addition to the extra `rollback()` interface, the transaction object is basically the same as `DbClient`. The transaction class is `Transaction`. For details of the `Transaction` class, see Transaction.

ORM

Drogon also provides support for **ORM**. Users can use the drogon_ctl command to read the tables in the database and generate the corresponding model source code. Then, execute the database operations of

these models through the `Mapper<MODEL>` class template. Mapper provides simple and convenient interfaces for standard database operations, allowing users to make the additions, deletions, and changes to the table without writing sql statements. For **ORM**, please refer to ORM

Next: DbClient