



Distribution using mongodb servers for storage and retrieval of data

Table of Contents

1	Synopsis
2	Description
2.1	Modules
2.1.1	MongoDB/Client.pm6
2.1.2	MongoDB/DataBase.pm6
2.1.3	MongoDB/Collection.pm6
2.1.4	MongoDB/Cursor.pm6
2.1.5	MongoDB/Log.pm6
2.1.6	MongoDB/Uri.pm6
2.1.7	MongoDB/HL/Users.pm6
2.1.8	MongoDB/HL/Collection.pm6
2.1.9	MongoDB/Server.pm6
2.1.10	MongoDB/Server/Socket.pm6
2.1.11	MongoDB/Server/Control.pm6
2.1.12	MongoDB/Server/Monitor.pm6
2.1.13	MongoDB/Wire.pm6
2.1.14	MongoDB/Header.pm6
2.1.15	MongoDB/MDBConfig.pm6
2.1.16	MongoDB.pm6
3	MongoDB
3.1	Enumerations
3.1.1	TopologyType
3.1.2	ServerStatus
3.1.3	WireOpcode
3.1.4	QueryFindFlags
3.1.5	ResponseFlags
3.2	Constants
3.2.1	C-MAX-SOCKET-UNUSED-OPEN
3.3	Subsets
3.3.1	PortType
3.3.2	Helper subsets when module cannot be loaded creating circular dependencies

Synopsis

```

use MongoDB::Client;
use MongoDB::Database;

use BSON::Document;

my MongoDB::Client $client .= new(:uri('mongodb://'));
my MongoDB::Database $database = $client.database('myPetProject');

# Inserting data
my BSON::Document $req .= new: (
  insert => 'famous-people',
  documents => [
    BSON::Document.new((
      name => 'Larry',
      surname => 'Wall',
    )),
  ]
);

my BSON::Document $doc = $database.run-command($req);
if $doc<ok> {
  say "Pffoei, inserted";
}

```

Description

This distribution provides a set of modules which can help you accessing mongod server or servers.

Modules

Below are the modules provided with the distribution. The top few are the most imported ones. These will help to connect to a server, define the database and collection and to enter or modify documents in the collections. The core operation is [run-command](#) from [MongoDB::Database](#).

MongoDB/Client.pm6

This module defines class [MongoDB::Client](#) and is used to connect to mongod and mongos servers. This module can create a [MongoDB::Database](#) object for you as well as a [MongoDB::Collection](#) object.

MongoDB/DataBase.pm6

Definition of class [MongoDB::DataBase](#). Its main task is to provide the method [run-command](#) to access data, modify a server or to get information. A [MongoDB::Collection](#) object can be created using Database.

MongoDB/Collection.pm6

Defines class [MongoDB::Collection](#). The main purpose of this module is to help to find data in a collection. Inserting or modifying documents in a collection is mainly done using [run-command](#) which uses the collection name in the command. In those cases a collection object is not needed

MongoDB/Cursor.pm6

Defines class [MongoDB::Cursor](#). The module is used to iterate through found data searched for by [find\(\)](#) from the [MongoDB::Collection](#) module. The object is almost never generated by the user but is returned by the find method.

MongoDB/Log.pm6

Logging module. Will be changed later in some external module.

MongoDB/Uri.pm6

Class `MongoDB::Uri` used by `MongoDB::Client` to parse the uri string.

MongoDB/HL/Users.pm6

Defines class `MongoDB::HL::Users` to administer user accounts. It is placed in HL (higher level) because all operations could be done with just the run-command. This module adds facilities like checking password length etc.

MongoDB/HL/Collection.pm6

Defines class `MongoDB::HL::Collection` to do more than the lower level Collection provides.

MongoDB/Server.pm6

Class `MongoDB::Server` is a class to represent a mongodb server. This class can return Socket objects. It also authenticates when necessary. This is not to be used directly.

MongoDB/Server/Socket.pm6

Class `MongoDB::Server::Socket` to represent a connection to a server. This is not to be used directly.

MongoDB/Server/Control.pm6

Class `MongoDB::Server::Control` is used to start and stop a mongodb server using configuration files. Mostly needed for tests and is not to be used directly.

MongoDB/Server/Monitor.pm6

Class `MongoDB::Server::Monitor` is used to monitor a mongodb server for changes in the server state. Not to be used directly.

MongoDB/Wire.pm6

Defines class `MongoDB::Wire`. Module is used to send and receive data. Not to be used directly.

MongoDB/Header.pm6

Module `MongoDB::Header` helps `MongoDB::Wire` to encode and decode documents for transport to and from the mongodb server. This is not to be used directly.

MongoDB/MDBConfig.pm6

Class `MongoDB::MDBConfig` is a config database singleton. This is not to be used directly.

MongoDB.pm6

Module to hold basic information reachable from other modules. Further it defines a role and an Exception class. See for more below.

MongoDB

```
package MongoDB { ... }
```

Base module to hold constants, enums and subs

Enumerations

TopologyType

```
enum TopologyType is export <
  C-UNKNOWN-TPLGY C-STANDALONE-TPLGY C-REPLSET-WITH-PRIMARY-TPLGY
  C-REPLSET-NO-PRIMARY-TPLGY
>;
```

Used to describe the state of the Client object. See also [MongoDB::Client](#).

ServerStatus

```
enum ServerStatus is export <
  C-UNKNOWN-SERVER C-NON-EXISTENT-SERVER C-DOWN-SERVER C-RECOVERING-SERVER      = 13;  #
  C-REJECTED-SERVER C-GHOST-SERVER

  C-REPLICA-PRE-INIT C-REPLICASET-PRIMARY C-REPLICASET-SECONDARY
  C-REPLICASET-ARBITER

  C-SHARDING-SERVER C-MASTER-SERVER C-SLAVE-SERVER
>;
```

Used to describe the status of a Server object. See also [MongoDB::Client](#).

WireOpcode

```
enum WireOpcode is export (
  :C-OP-REPLY(1),
  :C-OP-MSG(1000), :C-OP-UPDATE(2001), :C-OP-INSERT(2002),
  :C-OP-RESERVED(2003), :C-OP-QUERY(2004), :C-OP-GET-MORE(2005),
  :C-OP-DELETE(2006), :C-OP-KILL-CURSORS(2007),
);
```

Operational codes to transport data to or from the mongodb server. The codes are used internally.

QueryFindFlags

```
enum QueryFindFlags is export (
  :C-NO-FLAGS(0x00), :C-QF-RESERVED(0x01),
  :C-QF-TAILABLECURSOR(0x02), :C-QF-SLAVEOK(0x04),
  :C-QF-OPLOGREPLAY(0x08), :C-QF-NOCURSOR TIMEOUT(0x10), :C-QF-AWAITDATA(0x20),
  :C-QF-EXHAUST(0x40), :C-QF-PORTAIL(0x80),
);
```

Flags to be used with e.g. find(). See also [MongoDB::Collection](#).

ResponseFlags

```
enum ResponseFlags is export (
  C-RF-CursorNotFound(0x01), C-RF-QueryFailure(0x02),
  C-RF-ShardConfigStale(0x04), C-RF-AwaitCapable(0x08),
);
```

Response flags found in result documents from the server. Used internally.

Constants

C-MAX-SOCKET-UNUSED-OPEN

```
constant C-MAX-SOCKET-UNUSED-OPEN is export = 900;
```

Time in seconds that a socket can be left open unused.

Subsets

PortType

```
subset PortType of Int is export where 0 < $_ <= 65535;
```

Port type is a number denoting a specific protocol, e.g. 80 is for [http](#) and 27017 is for [mongodb](#). Any protocol served by a server can be given a different port than its default. The range is from 0 to 65535 where 0 to 1023 can only be used by servers with privileges.

Helper subsets when module cannot be loaded creating circular dependencies

```
subset ClientType is export where .^name eq 'MongoDB::Client';  
subset DatabaseType is export where .^name eq 'MongoDB::Database';  
subset CollectionType is export where .^name eq 'MongoDB::Collection';  
subset ServerType is export where .^name eq 'MongoDB::Server';  
subset SocketType is export where .^name eq 'MongoDB::Socket';
```

These types can be used when an object is provided in some call interface. Only to be used internally because you cannot create a new object from it.

Generated using Pod::Render, Pod::To::HTML, wkhtmltopdf