Perl 6 MongoDB driver

Marcel Timmerman <mt1957@gmail.com>

Abstract

MongoDB is a *Non SQL* database which uses *Binary JSON (BSON)* to store and load information in a database. With the mongodb package a shell program called mongo is available to give instructions to a mongodb server.

To work with data on the server from within a program a driver is needed. There are drivers for many program languages. This document describes a driver for the Perl6 language. In the perl6 ecosystem, which might grow into a cpan like system later, there are two packages needed to work with the driver. These are *MongoDB* and BSON. BSON is automatically installed with other necessary modules.

The latest version of this document is generated on date 2017-01-31

Table of Contents

1. Introduction

The purpose of this document is to show how to work with the perl6 mongodb driver and not about how to design your database among other things. There are plenty of good books and documents out there, not to mention, the mongodb website. A few things need to be repeated in this document however.

There are quite a few modules written to perform the tasks at hand but not all modules will be explained here because many of them are modules defining classes to be used in the background and are not used by applications directly.

Furthermore, this document is not a reference. There are other documents for that, written to document the attributes, (sub)methods and subs in a class. There will be a list of references at the end of the document.

This document assumes that the reader is aware of at least the basics of the mongodb database and what one can do with it. Also some perl 6 knowledge will be necessary.

As a last remark, the driver is still in development. Although many parts are accomplished, some parts still need to be implemented like authentication agains kerberos or LDAP. Furthermore, there are some improvements needed to speedup the operations.

The following sections will be explained:

- *Implementation*.
  - *BSON::Document*. This is the basic vehicle to insert, update retrieve and send commands to the database server. In this section there is an explanation of the supported types as well as different ways to make requests. Some detailed perl6 is necessary to understand mistakes often made when creating the data structures.
  - *URI*. The URI tells the software how to connect to a server and select the proper server.
  - *MongoDB::Client*. This module is the starting point of all applications which need access to a mongodb database server.
  - *MongoDB::Database*.
  - *MongoDB::Collection*.
  - *MongoDB::Cursor*.
  - *MongoDB::Server*.
  - *MongoDB::Server::Control*.
- *Dependensies*. There are some dependencies which are explained a bit here. These are e.g. the server and its version, modules like BSON, PKCS5, Auth::SCRAM etcetera.
- *Examples*. Of course, a document whithout examples is a bit like an empty box as a present.

2. Implementation

2.1. BSON::Document

2.2. URI

2.3. MongoDB::Client

2.3.1. Server states and topology

Table 1. Server states depending on isMaster outcome

Server state	isMaster command result
SS-Unknown	Initial, or after a network error or failed ismaster call, or "ok: 1" not in ismaster response.
SS-Standalone	No "msg: isdbgrid", no setName, and no "isreplicaset: true".
SS-Mongos	"msg: isdbgrid"
SS-PossiblePrimary	Not yet checked, but another member thinks it is the primary.
SS-RSPrimary	"ismaster: true", "setName" in response.
SS-RSSecondary	"secondary: true", "setName" in response.
SS-RSArbiter	"arbiterOnly: true", "setName" in response.
SS-RSOther	"setName" in response, "hidden: true" or not primary, secondary, nor arbiter. E.g. starting up or recovering.
SS-RSGhost	"isreplicaset: true" in response. E.g. briefly during server startup, in an uninitialized replica set, or when the server is shunned (removed from the replica set config).

Table 2. Topology controlled by server states

Topology type	Server states
TT-Unknown	When a deployment has this topology type, no servers are suitable for read or write operations. These are servers which did not respond on initial connection or threw an exception because of e.g. a DNS lookup failure. All server states of these servers herded by the Client object is SS-Unknown.
TT-Single	A deployment of topology type TT-Single contains only a single server of any state except SS-Unknown. This topology type signifies a direct connection intended to receive all read and write operations.
TT-Sharded	A deployment of topology type TT-Sharded contains one or more servers of type SS-Mongos or SS-Unknown of at least one is SS-Mongos.
TT-ReplicaSet-NoPrimary	A deployment with this topology type can have a mix of server types: SS-RSSecondary, SS-RSArbiter, SS-RSOther, SS-RSGhost, SS-Unknown or SS-PossiblePrimary.
TT-ReplicaSet-WithPrimary	A deployment with this topology type can have a mix of server types: SS-RSPrimary, SS-RSSecondary, SS-RSArbiter, SS-RSOther, SS-RSGhost, SS-Unknown or SS-PossiblePrimary.

2.3.2. Making a connection

2.3.3. Server selection

- Record the server selection start time
- If the topology wire version is invalid, raise an error
- Find suitable servers by topology type and operation type
- If there are any suitable servers, choose one at random from those within the latency window and return it; otherwise, continue to step
- Request an immediate topology check, then block the server selection thread until the topology changes or until the server selection timeout has elapsed
- If more than serverSelectionTimeoutMS milliseconds have elapsed since the selection start time, raise a server selection error
- Goto Step

2.4. MongoDB::Database

2.4.1. run-command()

2.5. MongoDB::Collection

2.5.1. find()

2.6. MongoDB::Cursor

2.6.1. fetch()

2.6.2. iterating over documents

2.7. MongoDB::Server

2.8. MongoDB::Server::Control

3. Dependencies

3.1. BSON

3.2. MongoDB server

4. Examples

4.1. Starting and stopping a server using the configuration

This method, using a configuration file, is also used to test the modules to help starting and stopping a locally installed server. There are several steps in order to configure it properly.

- *Configuration file*.
- *Server selection*.
- *Starting and stopping*.

4.1.1. Configuration file

4.1.2. Server selection

4.1.3. Starting and stopping

4.2. Making a replica server

4.2.1. Preparing

4.2.2. Initializing

5. References to books, websites, articles and pod-documents

5.1. Web Pages

MongoDB Manual covering all aspects of what is possible. Source is from MongoDB, Inc. EPub edition [http://docs.mongodb.com/master/MongoDB-manual.epub]

MongoDB Driver Glossary and References

B

Binary JSON	BSON is a computer data interchange format used mainly as a data storage and network transfer format in the MongoDB database. See also on WikipediA [https://nl.wikipedia.org/wiki/BSON].

J

JavaScript Object Notation	JavaScript Object Notation) is an open-standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs. See also on WikipediA [https://nl.wikipedia.org/wiki/JSON].

M

MongoDB	MongoDB (from humongous) is a free and open-source cross-platform document-oriented database program.

N

Non SQL	A NoSQL (originally referring to "non *Structured Query Language* ", "non relational" or "not only SQL" database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

S

Structured Query Language

SQL or Structured Query Language is a special-purpose domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS)

Index