NAME MongoDB - Access MongoDB server

SYNOPSIS

```
use MongoDB;

# Initialize
#
my $connection = MongoDB::Connection.new( );
my $database = $connection.database( 'test' );
my $collection = $database.collection( 'perl_users' );

# Insert documents
#
$collection.insert( %(name => 'Piet Hein', nick => 'ph', versions => [ 5, 6])
                  , %(name => 'Pietje Bell', nick => 'pb')
                  );

# Find everything
#
my $cursor = $collection.find( );
while $cursor.fetch( ) -> %document { %document.perl.say; }

# Or narrow down using condition.
#
$cursor = $collection.find( { nick => 'ph' } );
$cursor.fetch( ).perl.say;

# Update any document, watchout for the MongoDB commands which uses $'s
#
$collection.update( {}, {'$set' => {company => 'Dutch Corners'}});

# Update a specific document
#
$collection.update( {nick => 'ph'}, {'$set' => { company => 'Dutch Corners'}});

# Remove specific documents.
#
$collection.remove( { nick => 'ph' } );

# Remove all documents.
#
```

```
$collection.remove( );
```

DESCRIPTION

This set of modules will help you accessing a MongoDB server. All primitive functions are installed to insert, update, find and remove documents. http://docs.mongodb.org/meta-driver/latest/legacy/mongodb-wire-protocol/

METHODS

method insert (**@documents, Bool :$continue_on_error = False) {...}

Insert a document. You may specify more than one. These documents must all be hashes. Below are the possible ways to insert documents.

The flag **:continue_on_error** can be set to let the insert continue its job when a document fails.

```
my %d1 = k1 => 'v1', k2 => 'v2';
my $d2 = %(k1 => 'v1a');
$collection.insert( :continue_on_error, {%d1}, $d2);

my @docs = $%( k2 => 'v2a', k5 => 'v5'), $%( k1 => 'v1b', k2 => 'v2b');
$collection.insert(@docs);
```

method find (%query = { }, Int :$number_to_skip = 0, Int :$number_to_return = 0, Bool :$no_cursor_timeout = False) {...}

Find documents in the database. When query is empty all documents are returned, There are 2 options and a flag to control the search.

:number_to_skip is used to skip a number of documents.

:number_to_return is used to ask for a specific number of documents.

:no_cursor_timeout The server normally times out idle cursors after an inactivity period (10 minutes) to prevent excess memory use. Set this option to prevent that.

```
$cursor = $collection.find({nick => 'pb'});
$cursor.fetch( ).perl.say;
```

method update (%selector, %update, Bool :$upsert = False, Bool :$multi_update = False) {...}

Update documents in the database. There are 2 flags defined.

:upsert If set, the database will insert the supplied object into the collection if no matching document is found.

:multi_update If set, the database will update all matching objects in the collection. Otherwise only updates first matching doc.

The commands used by MongoDB such as $set, $inc and $push can easily create unexpected errors in perl programs because scalars are written the same way. Make sure you escape the $ sign or enclose the commands in single quoted strings to prevent interpolation.

```
# Update all documents
```

```
$collection.update({}, {'$set' => {company => 'Implix'}});

# Update documents for nick 'ph' or, when not existent, create a new document.
$collection.update( :upsert, {nick => 'pb'}, {'$push' => {versions => 7}});
```

method remove (%selector = {}, Bool :$single_remove = False) {...}

Remove the selected documents from the database.

:single_remove If set, the database will remove only the first matching document in the collection. Otherwise all matching documents will be removed.

```
# Remove first document for nick 'ph'.
$collection.remove( :single_remove, {nick => 'pb'});

# Remove all documents
$collection.remove();
```

DEPENDENCIES

Module MongoDB depends on Module BSON.

SEE ALSO

The MongoDB site at http://www.mongodb.org/

BUGS

AUTHORS

Paweł Pabian - Creator of the modules til january 2015
Marcel Timmerman - Maintainer since january 2015 (MARTIMM on github)

LICENSE AND COPYRIGHT