Monger, a Clojure client for MongoDB

- Home
- All guides
- API reference
- Community
- Code
- Change log
- More Clojure libraries
- Donate
- Clojure Docs

About this guide

This guide covers:

- Creating indexes with Monger
- Dropping indexes with Monger
- Creating a capped collection
- Using MongoDB TTL collections (MongoDB 2.2+)

What version of Monger does this guide cover?

This guide covers Monger 3.1 (including preview releases).

What version of MongoDB does this guide cover?

Some of the features covered in this guide require MongoDB 2.2.

Overview

MongoDB provides operations on collections that are not related to inserting, updating or querying data. Two examples of such operations are operations on indexes and dropping of a collection.

Indexes

Indexes in databases are data structures that allow for significant read query performance improvements on large data sets at the cost of a slight performance hit on write queries (because indexes need to be updated on writes).

MongoDB indexes are much like their relational data store counterparts: they are created for a particular collection and a set of fields, you can specify ordering on each field and they can be dropped or re-created.

Creating Indexes

To create an index on a collection, use `monger.collection/ensure-index`. It will create the index but only if it does not already exist:

```
(ns my.app
  (:require [monger.core :as mg]
            [monger.collection :as mc]))
```

```
(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")
      coll "documents"]
  ;; create an index
  ;; the index created as { :language 1 } will be named "language_1"
  (mc/ensure-index db coll "language_1")

  ;; create an index with a custom name
  ;; array-map produces a map that is ordered, which
  ;; is important for indexes
  (mc/ensure-index db coll (array-map :language 1) { :name "by-language" })

  ;; create a unique index
  (mc/ensure-index db coll (array-map :language 1) { :unique true })

  ;; create an index on multiple fields (will be automatically named language_1_created_at_1 by convention)
  (mc/ensure-index db coll (array-map :language 1 :created_at 1)))
```

`monger.collection/create-index` takes the same arguments but will fail if the index already exists.

Listing Indexes On a Collection

To get a set of indexes on a collection, use `monger.collection/indexes-on` function that takes a database collection name.

Dropping Indexes

To drop an index or all indexes on a collection, use `monger.collection/drop-index` and `monger.collection/drop-indexes`:

```
(ns my.app
  (:require [monger.core :as mg]
            [monger.collection :as mc]))

(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")]
  ;; drop all indexes from a collection
  (mc/drop-indexes db "events")

  ;; drop a specific index from a collection
  ;; language_1 refers to the index created as { :language 1 }
  (mc/drop-index db "documents" "language_1")

  ;; language_1_created_at_1 refers to the index created as { :language 1, :created_at 1 }
  (mc/drop-index db "documents" "language_1_created_at_1"))
```

Creating a Collection

To create a collection with Monger, use the `monger.collection/create` function like so:

```
(ns monger.docs.examples
  (:require [monger.core :as mg]
            [monger.collection :as mc]))

(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")]
  (mc/create db "recent_events" {}))
```

However, because MongoDB will create a collection automatically the first time you attempt to use it, it is usually not necessary to create collections except when you want them to be capped. This is what the next section covers.

Creating a Capped Collection

Capped collections in MongoDB are fixed sized collections that have a very high performance auto-FIFO age-out feature (age out is based on insertion order).

In addition, capped collections automatically, with high performance, maintain insertion order for the documents in the collection; this is very powerful for certain use cases such as logging or keeping "hot" data around for caching purposes.

To create a collection as capped, use the same `monger.collection/create` function with options:

```
(ns monger.docs.examples
  (:require [monger.core :as mg]
            [monger.collection :as mc]))

(defn- megabytes
  [^long n]
  (* n 1024 1024))

(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")]

  ;; creates a collection capped at 16 megabytes
  (mc/create db "recent_events" {:capped true :size (-> 16 megabytes)})

  ;; creates a collection capped at 1000 documents
  (mc/create db "recent_events" {:capped true :max 1000})
```

Using MongoDB TTL collections (MongoDB 2.2+)

MongoDB 2.2 and later versions support TTL (time-to-live, aka expiring) collections.

To enable document expiration, you need to create an index on a date field with the `:expireAfterSeconds` option:

```
(ns monger.docs.examples
  (:require [monger.core :as mg]
            [monger.collection :as mc]))

(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")]
  ;; expire documents with the `created-at` field value 120 seconds or more in the past.
  ;; expiration operation is performed about once a minute
  (mc/ensure-index db "recent_events" {:created-at 1} {:expireAfterSeconds 120})
```

Note that MongoDB runs the expiring thread (sometimes referred to as called `TTLMonitor`), about once per minute, so with TTL values less than 60 seconds this feature may be of little use.

Dropping a Collection

To drop a collection, use `monger.collection/drop` function that takes a database and collection name.

Renaming a Collection

To rename a collection, use `monger.collection/rename` that takes a database, old and new collection names.

What to Read Next

The documentation is organized as a number of guides, covering all kinds of topics.

We recommend that you read the following guides first, if possible, in this order:

- Integration with 3rd party libraries
- Map/Reduce
- GridFS support
- Using MongoDB Aggregation Framework
- Using MongoDB commands

Tell Us What You Think!

Please take a moment to tell us what you think about this guide on Twitter or the Monger mailing list

Let us know what was unclear or what has not been covered. Maybe you do not like the guide style or grammar or discover spelling mistakes. Reader feedback is key to making the documentation better.

comments powered by Disqus