MongoDB Shell

Summary: in this tutorial, you will learn about the mongo shell and how to use it to interact with the MongoDB database server.

Introduction to the mongo shell

The mongo shell is an interactive JavaScript interface to MongoDB. The mongo shell allows you to manage data in MongoDB as well as carry out administrative tasks.

> The mongo shell is similar to the mysql in MySQL, psql in PostgreSQL, and SQL*Plus in Oracle Database.

> Note that the `mongo` shell that is included in the MongoDB installation by default was deprecated in MongoDB v5.0. The `mongsh` is the replacement of the legacy `mongo` shell.

Before using the mongo shell, you need to download and install it (https://docs.mongodb.com/mongodb-shell/) .

To start the mongo shell, you open the Terminal on macOS and Linux or a Command Prompt on Windows and use the following command:

```
mongosh
```

> The mongo shell automatically connects to the MongoDB running on the local server (`localhost`) with the default port (`27017`). Therefore, you need to ensure that the MongoDB server is up and listening on this port before starting the mongo shell.

Note that you can use the mongo shell as a full-featured JavaScript interpreter and as a MongoDB client.

JavaScript interpreter

The mongo shell is a full-featured JavaScript interpreter so that you can execute any JavaScript code like this:

```
> Math.max(10,20,30);
30
```

The `mongo` shell allows you to enter multiline commands. It will detect if the JavaScript statement is complete when you press `enter` .

If the statement is not complete, the mongo shell will allow you to type on the next line after the three dots (`...`).

```
> function add(a, b) {
... return a + b;
... }
> add(10,20);
30
```

To clear the screen, you use the console.clear() like this:

```
console.clear()
```

MongoDB client

The mongo shell is a MongoDB client. By default, the mongo shell connects to the `test` database on the MongoDB server and assigns the database connection to the global variable called `db` .

The `db` variable allows you to see the current database:

```
> db
test
```

> Note that when you type a variable or an expression on the command line, the mongo shell will evaluate it and returns the result.

Besides supporting JavaScript syntax, the `mongosh` shell provides you with useful commands that easily interact with the MongoDB database server.

For example, you can use the `shows dbs` command to list all the databases on the server:

```
test> show dbs
admin         41 kB
config      73.7 kB
local       81.9 kB
```

The output shows three databases on the server.

To switch the current database to another, you use the `use <database>` command. For example, the following command switches the current database to the `bookdb` database:

```
test> use bookdb
switched to db bookdb
```

Note that you can switch to a database that does not exist. In this case, MongoDB will automatically create that database when you first save the data in it.

The mongo shell now assigns `bookdb` to the `db` variable:

```
> db
bookdb
```

Now, you can access the `books` collection from the `bookstore` database via the `db` variable like this:

```
> db.books
bookstore.books
```

Basic CRUD operations

The following section shows you how to create (C), read (R), update (U), and delete (D) a document. These operations are often referred to as CRUD operations.

> Note that this section only covers the basic CRUD operations (https://mongodbtutorial.org/mongodb-crud/) , you will learn about them in detail in the CRUD tutorial (https://mongodbtutorial.org/mongodb-crud/) .

Create

To add a document to a collection, you use the `insertOne()` method of the collection.

The following command adds a new document (or a new book) to the `books` collection:

```
db.books.insertOne({
   title: "MongoDB Tutorial",
   published_year: 2020
})
```

Output:

```
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f2f39fb82f5c7bd6c9375a8")
}
```

Once you press `enter`, the mongo shell sends the command to the MongoDB server.

If the command is valid, MongoDB inserts the document and returns the result. In this case, it returns an object that has two keys `acknowledged` and `insertedId`.

The value of the `insertedId` is the value of the `_id` field in the document.

When you add a document to a collection without specifying the `_id` field, MongoDB automatically assigns a unique `ObjectId` value to the `_id` field and add it to the document.

MongoDB uses the `_id` field to uniquely identify the document in the collection.

Read

To select documents in a collection, you use the `findOne()` method:

```
db.books.findOne()
```

Output:

```
{
  _id: ObjectId("62143f34cca1c7af0ad1d126"),
  title: 'MongoDB Tutorial',
  published_year: 2020
}
```

To format the output, you use the `pretty()` method like this:

```
db.books.find().pretty()
```

Output:

```
{
        "_id" : ObjectId("5f2f3d8882f5c7bd6c9375ab"),
        "title" : "MongoDB Tutorial",
        "published_year" : 2020
}
```

As you can see clearly from the output, MongoDB added the `_id` field together with other field-and-value pairs to the document.

Update

To update a single document, you use the `updateOne()` method.

The `updateOne()` method takes at least two arguments:

- The first argument identifies the document to update.
- The second argument represents the updates that you want to make.

The following shows how to update the `published_year` of the document whose title is `"MongoDB Tutorial"` :

```
db.books.updateOne(
        { title: "MongoDB Tutorial"},
        { $set: { published_year: 2019 }}
)
```

Output:

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

How it works.

The first argument identifies which document to update. In this case, it will update the first document that has the title `"MongoDB tutorial"` :

```
{title: "MongoDB Tutorial"}
```

The second argument is an object that specifies which fields in the document to update:

```
{
    $set: {
        published_year: 2019
    }
}
```

The `$set` is an operator that replaces a field value with a specified value. In this example, it updates the `published_year` of the document to `2019` .

Delete

To delete a document from a collection, you use the `deleteOne()` method. The `deleteOne()` method takes one argument that identifies the document that you want to delete.

The following example uses the `deleteOne()` method to delete a document in the `books` collection:

```
db.books.deleteOne({title: "MongoDB Tutorial"});
```

Output:

```
{
    "acknowledged": true,
    "deletedCount": 1
}
```

The output shows that one document has been deleted successfully via the `deletedCount` field.

To show all collections of the current database, you use the show collections command:

```
show collections
```

Output:

```
books
```

Currently, the `bookdb` database has one collection which is `books` .

Summary

- The mongo shell is an interactive JavaScript interface to MongoDB.
- The mongo shell allows you to manage data in MongoDB as well as perform administrative tasks.
- Use the `show dbs` command to list all databases on the server.
- Use the `use` command to switch to another database.
- Use the `show collections` to list all collections of the current database.
- CRUD is referred to as four basic functions including creating, reading, updating, and deleting data.