How To Connect Node.js to a MongoDB Database

Authored by: **ASPHostServer Administrator** [asphostserver@gmail.com]
Saved From: http://faq.asphosthelpdesk.com/article.php?id=157

Introduction

In this tutorial, we will show how to use Node.js to connect to a MongoDB database in a server and do some basic data manipulations.

Here are the following software components that will be used:

- Ubuntu 12.04 x32
- MongoDB v2.4.6
- Node.js v0.10.20
- The MongoDB Node.js driver

MongoDB

"MongoDB is an open source document-oriented database that provides high performance, high availability, and easy scalability"

If you are not familiar with MongoDB or don't have it installed, please check out this tutorial first.

Let's verify that the MongoDB process is running:

ps -ef | grep mongo

The output should look something like this:

mongodb 13071 0 02:27 ? 00:00:01 /usr/bin/mongod --config /etc/mongodb.conf

If it's not running, issue the following command from the MongoDB bin directory:

mongod

There is a console client that comes with MongoDB. To launch it, issue the following command:

mongo

You will see an output like this (you can ignore the warnings):

MongoDB shell version: 2.4.4

connecting to: test

Server has startup warnings:

Mon Oct 7 20:40:35.209 [initandlisten]

Mon Oct 7 20:40:35.209 [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000

>

Run this command to list the existing databases:

show dbs

Run this command to display the selected database:

db

Run the following command to switch to the "test" database and display the collections it contains:

use test

show collections

Here is a list of commands that you can use in the console client, you can get the full list of commands by typing "help":

show dbs #show database names

show collections #show collections in current database

show users # show users in current database

show profile # show most recent system.profile entries with time >= 1ms

show logs # show the accessible logger names

show log [name] # prints out the last segment of log in memory, 'global' is default

use <db_name> #set current database

db.foo.find() # list objects in collection foo

db.foo.find({ a : 1 }) #list objects in foo where a == 1

it #result of the last line evaluated; use to further iterate

exit #quit the mongo shell

Node.js

"Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."

If you don't have this installed, please take the time to follow the instructions in this tutorial first.

Let's verify that the Node.js process is running:

```
node -v
```

You should see the Node.js version as the command output.

The MongoDB Node.js Driver

This driver is the officially supported Node.js driver for MongoDB. It is written in pure JavaScript and provides a native asynchronous Node.js interface to MongoDB.

Use the node package manager "npm" to install the driver:

```
npm install mongodb
```

Connecting to MongoDB and Performing Data Manipulation

Now it is time to write the code that will allow your Node.js application to connect to MongoDB. Three operations will be covered: connecting, writing, and reading from the database.

To be able to execute your code, we will need to create a new file, we'll call it: 'app.js'.

Once you have the file, use your preferred editor to add the following code:

```
var MongoClient = require('mongodb').MongoClient

, format = require('util').format;

MongoClient.connect('mongodb://127.0.0.1:27017/test', function (err, db) {

if (err) {

throw err;

} else {

console.log("successfully connected to the database");

}

db.close();

});
```

Execute the app.js file by typing the following command:

```
node app.js
```

You should see the following string in the output: successfully connected to the database.

Now let's add some logic that inserts things to a new collection named "test_insertâ€•:

```
var MongoClient = require('mongodb').MongoClient

, format = require('util').format;
```

```
MongoClient.connect('mongodb://127.0.0.1:27017/test', function(err, db) {

if(err) throw err;

var collection = db.collection('test_insert');

collection.insert({a:2}, function(err, docs) {

collection.count(function(err, count) {

console.log(format("count = %s", count));

db.close();

});

});

});
```

Add another block of code that verifies that the data made it to the database:

```
var MongoClient = require('mongodb').MongoClient

, format = require('util').format;

MongoClient.connect('mongodb://127.0.0.1:27017/test', function(err, db) {

if(err) throw err;

var collection = db.collection('test_insert');

collection.insert({a:2}, function(err, docs) {

collection.count(function(err, count) {

console.log(format("count = %s", count));

});

});

// Locate all the entries using find

collection.find().toArray(function(err, results) {

console.dir(results);

// Let's close the db

db.close();

});
```

```
});
```

Congratulations! You now have the ability to connect, insert, and read data from your MongoDB database in a server using a Node.js application!