HOW TO USE MONGOOSE FOR MONGODB & NODEJS

Here we show how to use Mongoose to save data to a MongoDB.

(This article is part of our MongoDB Guide. Use the right-hand menu to navigate.)

What is Mongoose?

In this example, we will use **NodeJS**. Do not worry if you do not know NodeJS. We will explain every line of code. It is server-side JavaScript. It is a little difficult to understand at first, in part because it is asynchronous, meaning multi-threaded. So you have to use either:

- Synchronous functions only
- Callbacks

Otherwise when you do statement **n**, statement **n+1** will run right away too. So you can be, for example, processing a text file before you have read it completely.

Mongoose is an API on top of an API. It makes using the MongoDB NodeJS API easier to use.

Prerequisites

First, you need NodeJS version 6 or higher. Then install mongoose like this:

```
npm install mongoose
```

Sample Data

We will use data on smokers from the CDC (Center for Disease Control). Download that like this:

```
wget https://chronicdata.cdc.gov/views/wsas-xwh5
```

This dataset is a survey of smokers. We are only interested in the part of the data that shows which states have the most smokers (i.e., certain **cachedContents** JSON sections). That section looks like this:

```
{
"id" : 320867377,
"name" : "LocationDesc",
"dataTypeName" : "text",
"description" : "Location description",
"fieldName" : "locationdesc",
"position" : 3,
"renderTypeName" : "text",
"tableColumnId" : 20084901,
"width" : 109,
"cachedContents" : {
"largest" : "Wyoming",
"non_null" : 14069,
"null" : 0,
"top" : ,
smallest : String,
format : {
displayStyle : String,
align : String
}
}
});
```

We then compile the schema into a model like this:

```
var Smokers = mongoose.model('smokers', schema);
```

Read the JSON string data (**fs.readFileSync**) into a JSON JavaScript object (using **JSON.parse**). Notice that we use **fs.readFileSync** (i.e,. synchronous) instead of **fs.readFile** so that the next sections of code will wait until the read is complete.

```
fs = require('fs');

var d = fs.readFileSync('/home/walker/Documents/mongodb/tobacco.json',
'utf8', (err, data) => {
if (err) throw err;
return (d); });

var e = JSON.parse(d);
```

Next is the most complex part. The tobacco JSON file is two levels of JSON. So we have a loop

inside a look. Then we refer to **cachedContents** part of the JSON object using **e.columns.cachedContents**. We only want those records with **e.columns.cachedContents.top**, i.e., **top** is defined. (The other ones are geolocation data.)

We supply the **e.columns.cachedContents** to the constructor for the Smokers schema. Then we use the **save** method. Save runs asynchronously. But we don't care about that as it comes at the end of the program.

```
for (i in e) {
for (j in i) {

if (e.columns.dataTypeName != 'location') {
if (typeof(e.columns.cachedContents.top) != 'undefined') {
var smokers = new Smokers(e.columns.cachedContents);
console.log(e.columns.cachedContents);
smokers.save(function (err) {
if (err)return console.log(err);
})
}
}
}
}
```

The Complete Code

Copy and save this as loaddata.js. Then run it using: **node loaddata.js**. Notice there is no **main** or anything like that. It just runs top-to-bottom.

```
fs = require('fs');

var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/tobacco', { useMongoClient: true });
mongoose.Promise = global.Promise;

var schema = new mongoose.Schema({
cachedContents : {
largest : String,
non_null : Number,
null : Number,
top : ,
smallest : String,
format : {
displayStyle : String,
align : String
}
}
});

var d = fs.readFileSync('/home/walker/Documents/mongodb/tobacco.json',
'utf8', (err, data) => {
```

```
if (err) throw err;
return (d); });

var e = JSON.parse(d);

var Smokers = mongoose.model('smokers', schema);

for (i in e) {
for (j in i) {

if (e.columns.dataTypeName != 'location') {
if (typeof(e.columns.cachedContents.top) != 'undefined') {
var smokers = new Smokers(e.columns.cachedContents);
console.log(e.columns.cachedContents);
smokers.save(function (err) {
if (err)return console.log(err);
})
}
}
}
}
```

As saved record will look something like this:

```
{ largest: 'Wyoming',
non_null: 14069,
null: 0,
top:
,
smallest: 'Alabama' }
```

Tips

As you work with this you can delete and then show objects as shown below. **Remove** requires a filter. **{}** is a filter meaning all records. You use **db** instead of **tobacco** to denote the database. And **smokers** is the collection. And if you spell anything wrong it does not give you an error as it assumes you want to create a new object. (JavaScript is like that too as it has no error checking with regards to spelling variable names.)

```
sse tobacco
db.smokers.remove({})
db.smokers.find({}).pretty()
```

And then you can step through the data like this with the interactive node interpreter (i.e., run node and then paste in whatever code you want to study:

```
for (i in e) { for (j in i)
{"i=",i,"j=",j,console.log(e.columns.cachedContents) }}
```

Additional Resources

Mongoose: MongoDB object modelling for Node.js from Yuriy Bogomolov