

Node Program

MongoDB



Node.js version: 7
Last updated: Nov 2016

C.A.P. theorem

- >> **C**onsistency (strong vs. eventual-delay)
- >> **A**vailability
- >> **P**artition tolerance (on cluster)

SQL: C+A

No SQL!

- >> A+P from C.A.P.
- >> No relationships in the database.
- >> Redundancy is good.

NoSQL Databases

There are many types of NoSQL databases:

- >> Key-value stores (Redis, think hash tables)
- >> Document stores (mongoDB, think JSON)
- >> Columnar stores (hbase, average age)
- >> Graphs stores (neo4j)

SQL vs. NoSQL

Relation DB->normalized for any query, no biases

NoSQL->biases to specific query patterns that we have

MongoDB

MongoDB is a document store NoSQL database. It's great at distributed and scaling.

Launching MongoDB

Launch the mongod service with:

```
$ mongod
```

You should be able to see information in your terminal. The default port is 27017.

MongoDB Shell (mongo)

For the MongoDB shell, or mongo, launch in a new terminal window (let the server run), this command:

```
$ mongo
```


MongoDB Shell (mongo)

To test the database, use the JavaScript-like interface and commands save and find:

```
> db.test.save({a:1})
```

```
> db.test.find()
```

MongoDB uses JavaScript!

MongoDB Shell (mongo)

Useful MongoDB Shell commands:

```
>> > help
```

```
>> > show dbs
```

```
>> > use board
```

```
>> > show collections
```

```
>> > db.messages.remove();
```

MongoDB Shell (mongo)

Useful MongoDB Shell commands:

```
>> > var a=db.messages.findOne();
```

```
>> > print json(a);
```

```
>> > a.message="hi";
```

```
>> > db.messages.save(a);
```

```
>> > db.messages.find({});
```

MongoDB Shell (mongo)

Useful MongoDB Shell commands:

```
>> > db.messages.update( {name: "John"}, {$set:  
    {message: "bye"}});
```

```
>> > db.messages.find( {name: "John"} );
```

```
>> > db.messages.remove( {name: "John"} );
```

DEMO

MongoDB native driver vs. MongoDB Shell

MongoDB Native Driver (mongodb)

Node.js Native Driver for MongoDB (<https://github.com/christkv/node-mongodb-native>)

```
$ npm install mongodb --save
```

Establishing Connection

```
var MongoClient = require('mongodb').MongoClient
    , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
    assert.equal(null, err);
    console.log("Connected correctly to server");

    db.close();
});
```


Creating insertDocuments

```
var insertDocuments = function(db, callback) {  
  // Get the documents collection  
  var collection = db.collection('documents');  
  // Insert some documents  
  collection.insert([  
    {a : 1}, {a : 2}, {a : 3}  
  ], function(err, result) {  
    assert.equal(err, null);  
    assert.equal(3, result.result.n);  
    assert.equal(3, result.ops.length);  
    console.log("Inserted 3 documents into the document collection");  
    callback(result);  
  });  
}
```

Applying IntertDocuments

```
var MongoClient = require('mongodb').MongoClient
    , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
    assert.equal(null, err);
    console.log("Connected correctly to server");

    insertDocuments(db, function() {
        db.close();
    });
});
```

Updating Documents

```
var updateDocument = function(db, callback) {  
  // Get the documents collection  
  var collection = db.collection('documents');  
  // Update document where a is 2, set b equal to 1  
  collection.update({ a : 2 }  
    , { $set: { b : 1 } }, function(err, result) {  
    assert.equal(err, null);  
    assert.equal(1, result.result.n);  
    console.log("Updated the document with the field a equal to 2");  
    callback(result);  
  });  
}
```

Applying updateDocument

```
insertDocuments(db, function() {  
    updateDocument(db, function() {  
        db.close();  
    });  
});
```

Removing Documents

```
var removeDocument = function(db, callback) {  
  // Get the documents collection  
  var collection = db.collection('documents');  
  // Insert some documents  
  collection.remove({ a : 3 }, function(err, result) {  
    assert.equal(err, null);  
    assert.equal(1, result.result.n);  
    console.log("Removed the document with the field a equal to 3");  
    callback(result);  
  });  
}
```

Applying removeDocument

```
var MongoClient = require('mongodb').MongoClient
    , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
    assert.equal(null, err);
    console.log("Connected correctly to server");

    insertDocuments(db, function() {
        updateDocument(db, function() {
            removeDocument(db, function() {
                db.close();
            });
        });
    });
});
```

Finding Documents

```
var findDocuments = function(db, callback) {  
  // Get the documents collection  
  var collection = db.collection('documents');  
  // Find some documents  
  collection.find({}).toArray(function(err, docs) {  
    assert.equal(err, null);  
    assert.equal(2, docs.length);  
    console.log("Found the following records");  
    console.dir(docs);  
    callback(docs);  
  });  
}
```

Applying findDocuments

```
var MongoClient = require('mongodb').MongoClient
, assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  insertDocuments(db, function() {
    updateDocument(db, function() {
      removeDocument(db, function() {
        findDocuments(db, function() {
          db.close();
        });
      });
    });
  });
});
})
```


MongoDB BSON Data Types

Binary JSON, or BSON, it is a special data type which MongoDB utilizes. It is like to JSON in notation, but has support for additional more sophisticated data types.

<http://bsonspec.org>

BSON

Binary: the base64 representation of a binary string

Date: a 64-bit integer of the ISO-8601 date format with a mandatory time zone field following the template YYYY-MM-DDTHH:mm:ss.mmm<+/-Offset>

MongoDB BSON Data Types

Timestamp: a 64 bit value

OID: a 24-character hexadecimal string

DB Reference

MinKey

MaxKey

NumberLong: a 64 bit signed integer

Questions and Exercises



Native Driver Alternatives

- Mongoose: asynchronous JavaScript driver with optional support for modeling
- Mongoose: asynchronous JavaScript driver with optional support for modeling
- Mongolia: lightweight MongoDB ORM/driver wrapper
- Monk: Monk is a tiny layer that provides simple yet substantial usability improvements for MongoDB usage within Node.js

Workshop



```
$ [sudo] npm install -g learnyoumongo
```