Dynamic Web Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

PART 3

mongoose & mongoDB

Table of Contents

mongoose

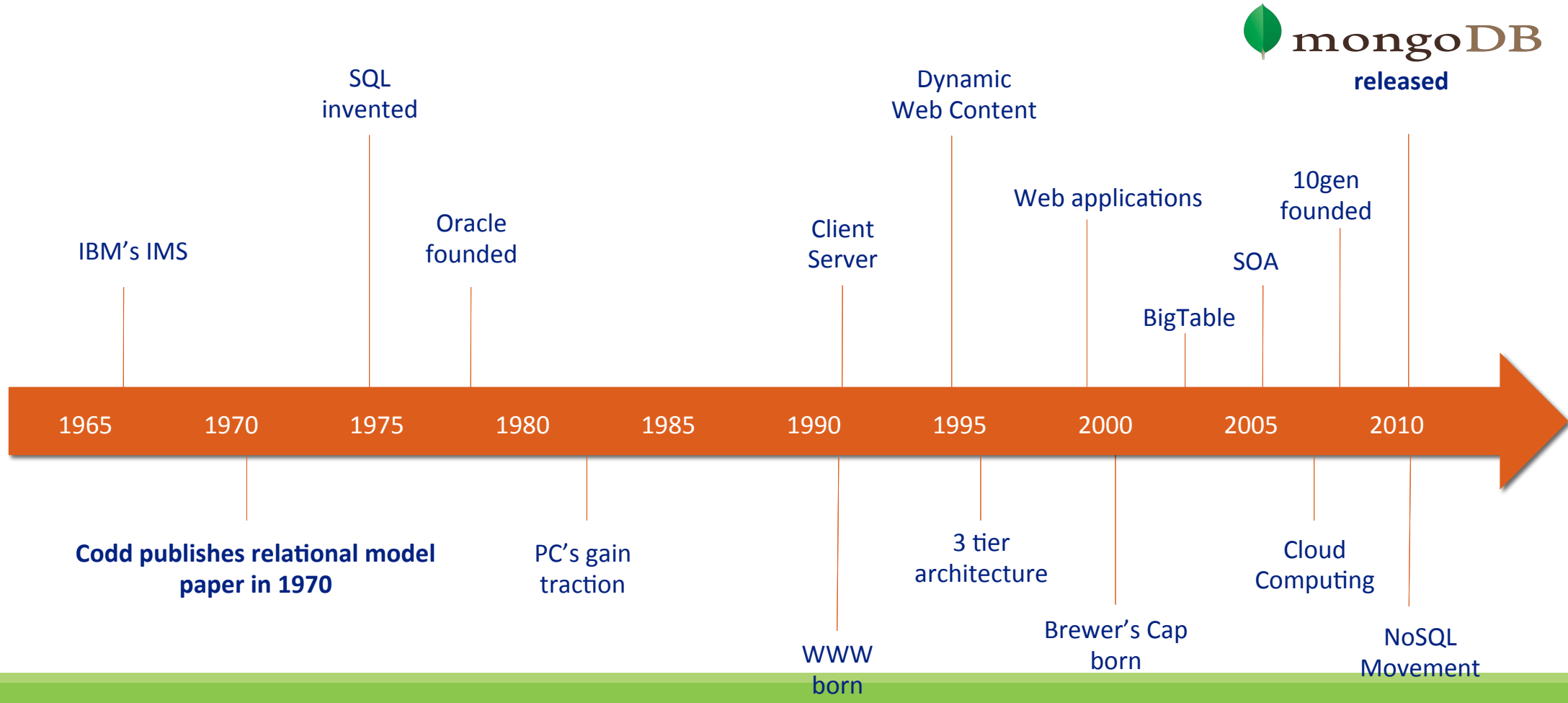
Databases

HISTORY & BACKGROUND

mongoose

mongoDB

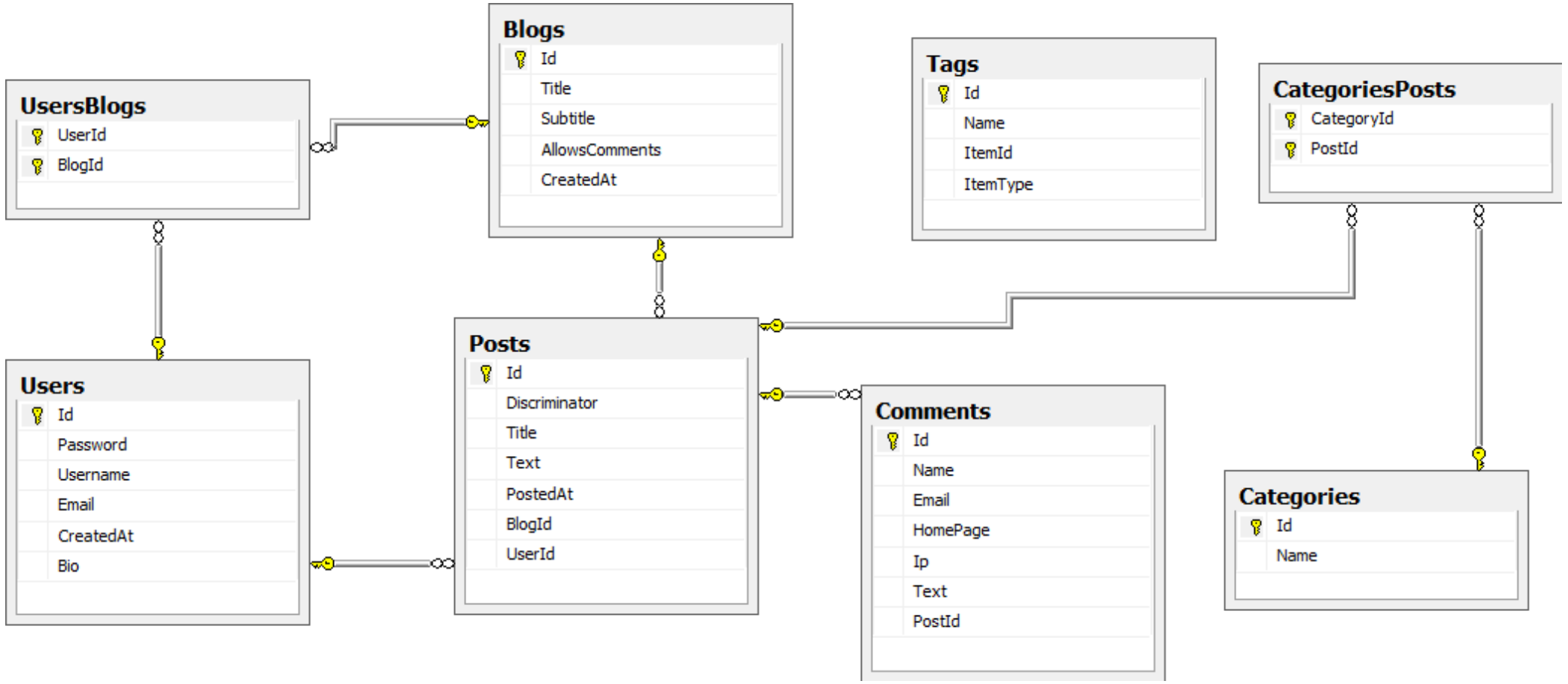
Dawn of Databases to Present



mongoose

mongoDB

Relational Databases



mongoose

Relational Databases Strengths

Data stored in a RDBMS is very compact (disk was more expensive)

SQL and RDBMS made queries flexible with rigid schemas

Rigid schemas helps optimize joins and storage

Massive ecosystem of tools, libraries and integrations

Been around 40 years!

mongoose

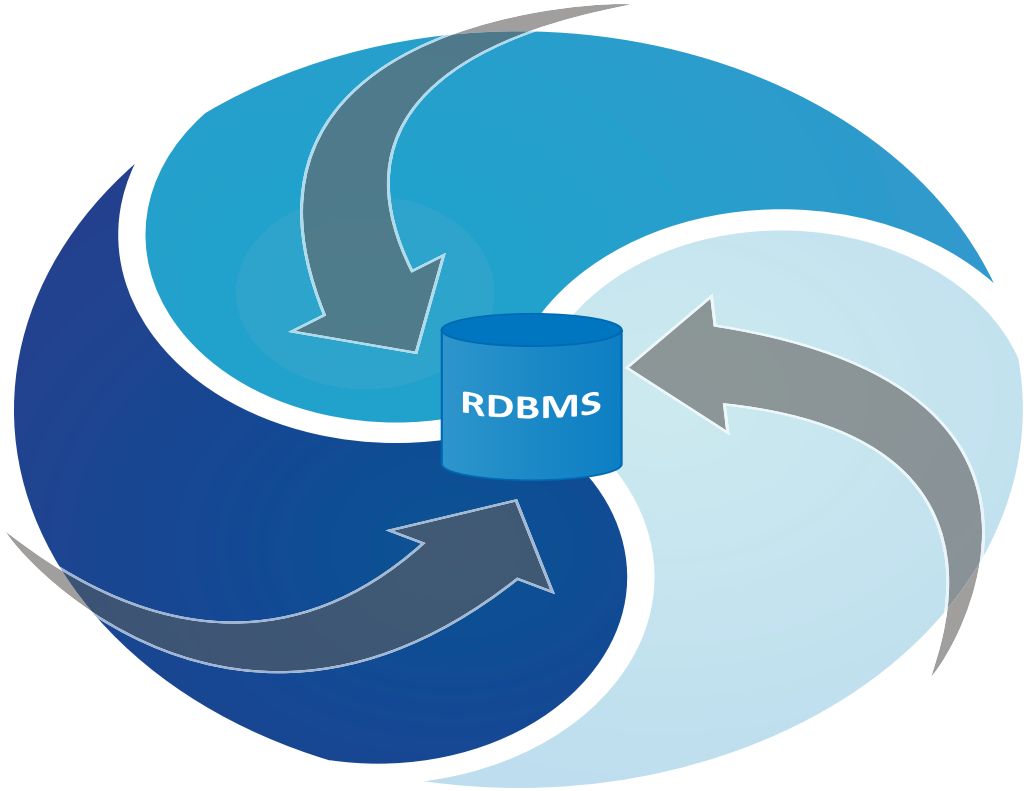
mongoDB

Enter Big Data

Gartner uses the 3Vs to define

- Volume - Big/difficult/extreme volume is relative
- Variety
  - Changing or evolving data
  - Uncontrolled formats
  - Does not easily adhere to a single schema
  - Unknown at design time
- Velocity
  - High or volatile inbound data
  - High query and read operations
  - Low latency

RDBMS Challenges



DATA VARIETY & VOLATILITY

- Extremely difficult to find a single fixed schema
- Don't know data schema a-priori

VOLUME & NEW ARCHITECTURES

- Systems scaling horizontally, not vertically
- Commodity servers
- Cloud Computing

TRANSACTIONAL MODEL

- N x Inserts or updates
- Distributed transactions

mongoose

mongoDB

Enter NoSQL

Modern NoSQL theory and offerings started in early 2000s
(idea around much longer)

Modern usage of term introduced in 2009

NoSQL = Not Only SQL

A collection of very different products

Alternatives to relational databases when they are a bad fit

Motives

- Horizontally scalable
- Flexibility

mongoose

NoSQL Databases – Key-Value

Key-Value Stores

- Key-Value Stores
- Value (Data) mapped to a key (think primary)

Big Table Descendants

- Looks like a distributed multi-dimension map
- Data stored in a column oriented fashion
- Predominantly hash based indexing

Document oriented stores

- Data stored as either JSON or XML documents

10gen & MongoDB

2007

- Eliot Horowitz & Dwight Merriman tired of reinventing the wheel
- 10gen founded
- MongoDB Development begins

2009

- Initial release of MongoDB

73M+ in funding

- Funded by Sequoia, NEA (New Enterprise Associates), Union Square Ventures, Flybridge Capital

MongoDB

OBJECT-DOCUMENT MODEL

Overview: MongoDB – What is it?

MongoDB is an open-source **document store** database

- The leading NoSQL database (see next slide)
- Save JSON-style objects with dynamic schemas

Support many features:

- Support for indices
- Rich, document-based queries (CRUD operations)
- Flexible aggregation and data processing
- Store files of any size without complicating your stack
- Fast in-place updates

mongoose

DB-Engines Ranking (Dec 2015)

289 systems in ranking, December 2015

Rank			DBMS	Database Model	Score		
Dec 2015	Nov 2015	Dec 2014			Dec 2015	Nov 2015	Dec 2014
1.	1.	1.	Oracle	Relational DBMS	1497.55	+16.61	+37.76
2.	2.	2.	MySQL	Relational DBMS	1298.54	+11.70	+29.96
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1123.16	+0.83	-76.89
4.	4.	↑ 5.	MongoDB	Document store	301.39	-3.22	+54.87
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	280.09	-5.60	+26.09
6.	6.	6.	DB2	Relational DBMS	196.13	-6.40	-14.13
7.	7.	7.	Microsoft Access	Relational DBMS	140.21	-0.75	+0.31
8.	8.	↑ 9.	Cassandra	Wide column store	130.84	-2.08	+36.78
9.	9.	↓ 8.	SQLite	Relational DBMS	100.85	-2.60	+6.15
10.	10.	10.	Redis	Key-value store	100.54	-1.87	+12.66

mongoose

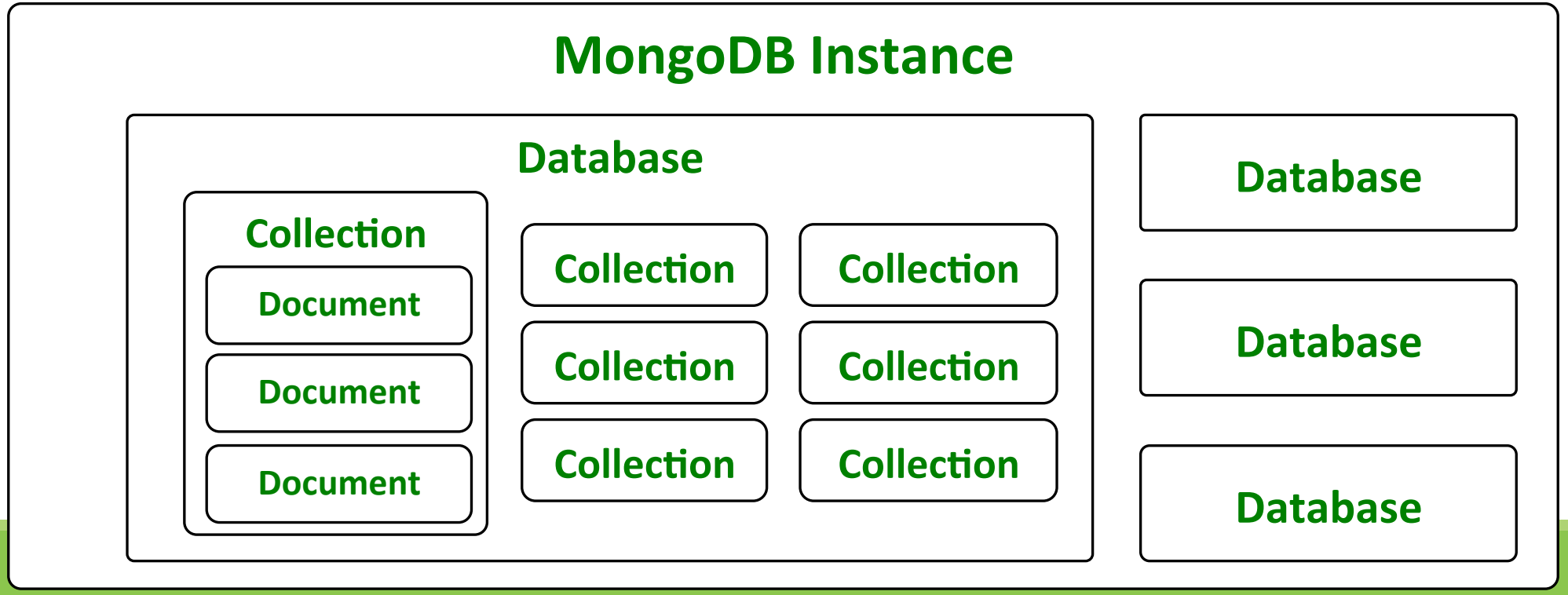
Overview: MongoDB Key Features

- Data stored as documents (JSON)
- Schema free
- CRUD operations – (Create Read Update Delete)
- Atomic document operations
- Ad hoc Queries like SQL
  - Equality
  - Regular expression searches
  - Ranges
  - Geospatial
- Secondary indexes
- Sharding (sometimes called partitioning) for scalability
- Replication – HA and read scalability

Overview: MongoDB Data Model

A MongoDB instance can have many databases

- A database can have many collections
  - A collection can have many documents



Overview: MongoDB Documents

Documents in MongoDB are JSON objects

Module Name: Databases

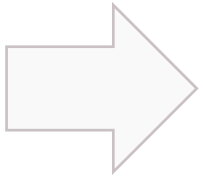
Students: 250

Titles:

- Intro to SQL
- NoSQL databases
- Entity Framework

Evaluation:

- Homework: 10
- Exam: 60
- Teamwork: 20
- Attendance: 10



```
{
 ”module_name": "Databases",
 "students": 250,
 "titles": [
   "Intro to SQL",
   "NoSQL databases",
   "DB Performance"
 ],
 "evaluation": {
   "homework": 10,
   "exam": 60,
   "teamwork": 20,
   "attendance": 10
}
```

Overview: MongoDB Documents

MongoDB documents can have many types of values

- Numbers, Booleans
- Strings, Object Ids
- Dates
- Objects
  - Also know as nested documents
- Arrays

```
17, 3.14, true/false
```

```
"David Drohan"
```

```
2014-09-01T14:58:48.126Z
```

```
{
 "username": ”daveyd",
 "accessLevel": 2
}
```

```
['bananas', 'oranges']
```

MongoDB Installation

INSTALLING & USING MONGODB

MongoDB Installation

Download MongoDB from the official web site:

- https://www.mongodb.org/downloads
- Installers for all major platforms

How to start MongoDB?

- Go to installation folder and run **mongod**
- Paths need some configuration

```
$ cd C:\MongoDb\bin
```

  - You should specify databases path
  - or use default path and create folder **C:\data\db**

```
$ mongod --dbpath D:\mongodb\data
```

```
$ mongod
```

How to use MongoDB

Connect to a **mongodb**	`$ mongo`
Select a database	`$ use <database name>`
Show database name	`$ db`
Show all databases	`$ show dbs`

Authentication

How to create users in a database?

```
$ use ssd4db
$ db.createUser({
  "user": "daveyd",
  "pwd": "ssd412345",
  "roles": ["readWrite", "dbAdmin"]
})
```

Database user roles:

- **read**, **readWrite**, **dbAdmin**, **dbOwner**, **userAdmin**
- **userAdminAnyDatabase**

Login:

```
db.auth("daveyd", "ssd412345")
```

{ name: mongo, type: DB }

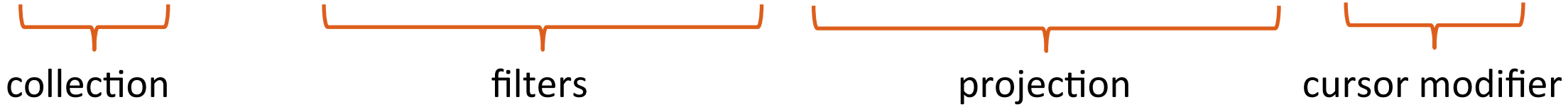
MongoDB Queries

INSTALLING & USING MONGODB

MongoDB Queries – Read

Read query

```
$ db.courses.find({ level: { $gt: 2 }}, { module_name: true }).limit(5);
```



Same as (in SQL):

```
SELECT module_name
FROM courses
WHERE level > 2
LIMIT 5
```

MongoDB Queries – Read (2)

```
$ db.courses.find({
    module_name: ”Databases"
  });
```

```
$ db.courses.find({
    $or: [{ level: { $gt: 1 } }, { level: { $lt: 3 } }]
  });
```

Operators:

- Comparison – **$gt**, **$gte**, **$et**, **$lt**, **$lte**
- Logical – **$and**, **$nor**, **$not**, **$or**
- Element – **$exists**, **$type**

MongoDB Queries – Insert

Insert query

```
$ db.courses.insert({
    module_name: "Databases",
    description: "Databases description",
    level: 3
  })
```

Same as (in SQL):

```
INSERT INTO courses (module_name, description, level)
VALUES ("Databases", "Databases description", 3)
```

MongoDB Queries – Bulk Insert

Inserting bulk data in MongoDB

```
var bulk = db.courses.initializeUnorderedBulkOp();
bulk.insert( { name: "ASP.NET MVC", level: 3 } );
bulk.insert( { item: "Web Development", level: 3 } );
bulk.insert( { item: "SPA Applications", level: 2 } );
bulk.execute();
```

MongoDB Queries – Update

Update query

```
$ db.courses.update(
  { module_name: { $et: "Databases" }},
  { $set: { description: "Concepts, MSSQL, MySQL, Oracle"}},
  { multi: true }
);
```

Same as (in SQL):

```
UPDATE courses
SET description = "Concepts, MSSQL, MySQL, Oracle"
WHERE module_name = "Databases"
```

MongoDB Queries – Delete

Delete query

```
$ db.courses.remove({ module_name: "Databases" });
```

Same as (in SQL):

```
DELETE FROM courses
WHERE module_name = "Databases"
```

MongoDB Queries – Aggregation

Aggregation is operation that process data records and return computed results

```
$ db.courses.aggregate([
  { $match: { isActive: true } },
  { $group: { level: "$level", students: { $sum: "$studentsCount" }} }
])
```

Result:

```
[
  { level: 1, students: 460 },
  { level: 2, students: 350 },
  { level: 3, students: 200 }
]
```

MongoDB Management Tools

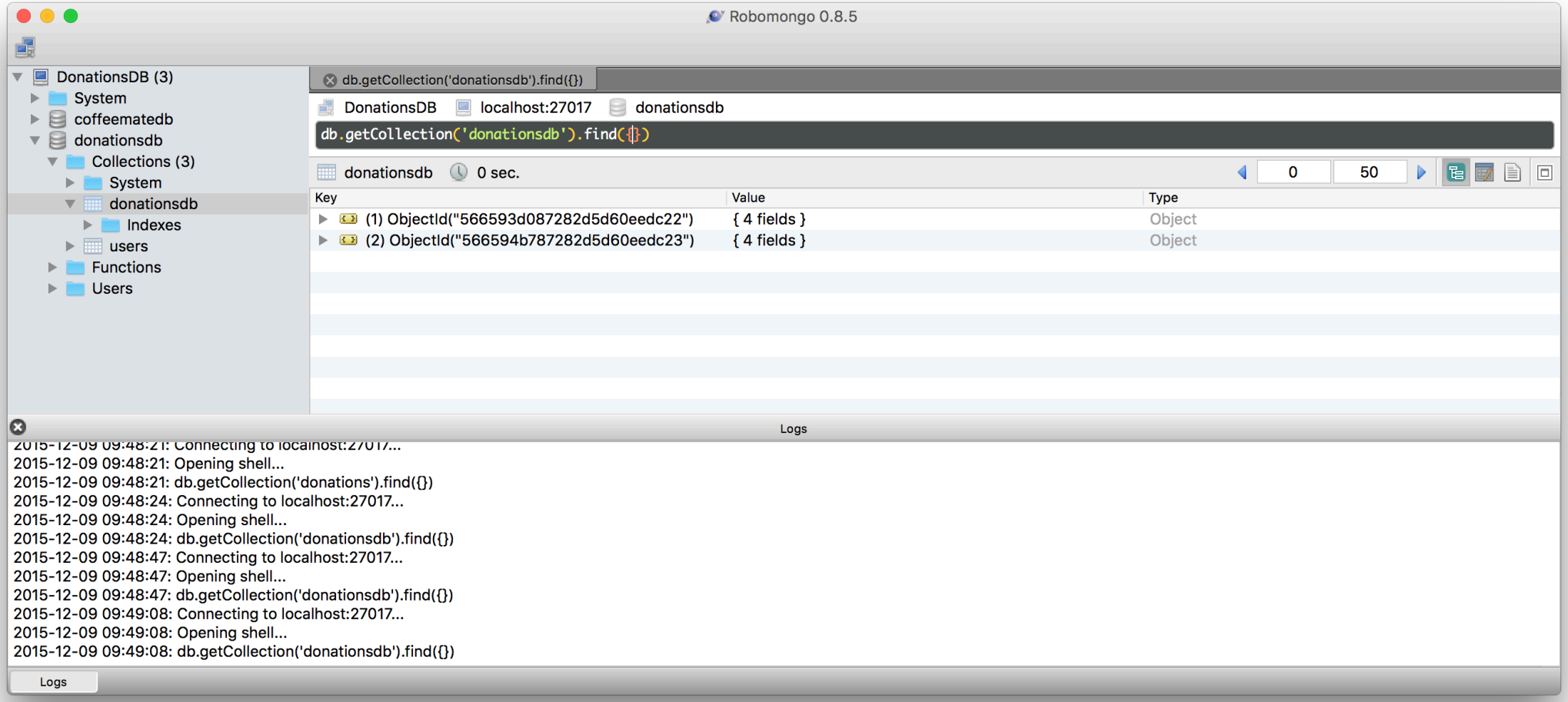
MongoDB is an open-source DB system

- So there are many available viewers

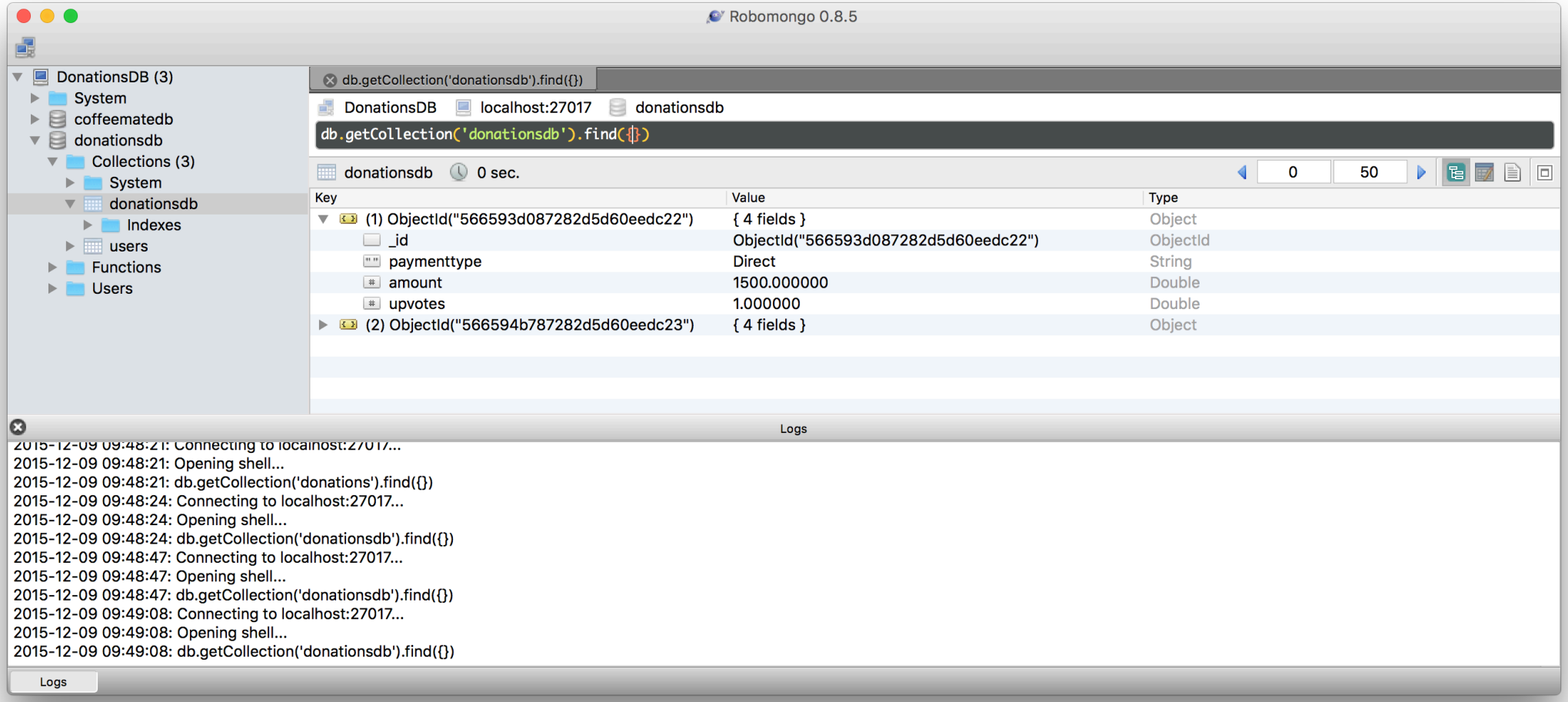
Some, but not all are:

- MongoDB CLI
  - Comes with installation of MongoDB
  - Execute queries inside the CMD/Terminal
- MongoVUE & UMongo & Robomongo
  - Provide UI to view, edit are remove DB documents
  - Execute queries inside the tool

MongoDB - Robomongo



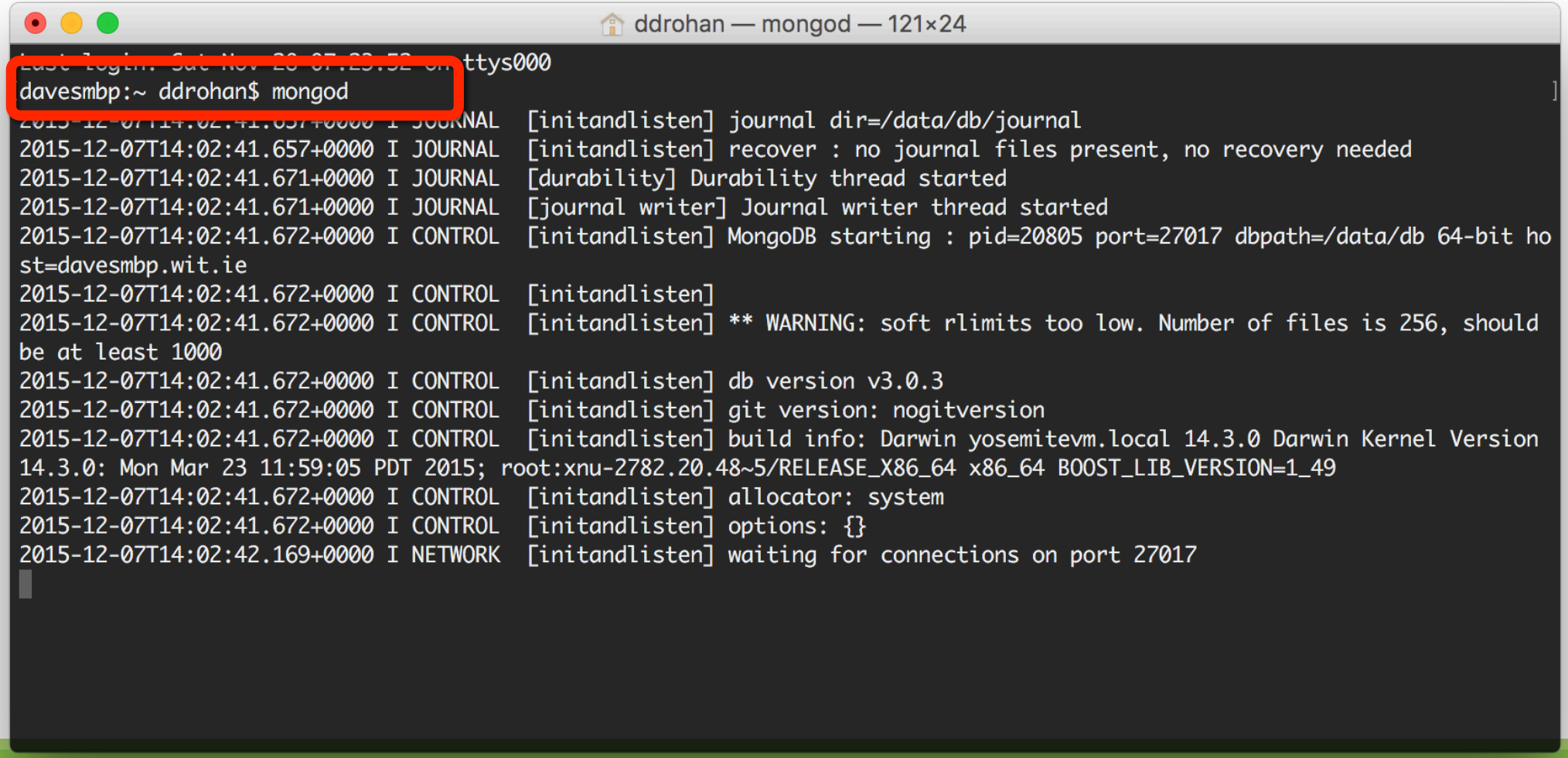
MongoDB - Robomongo



Donationweb

CASE STUDY EXAMPLE

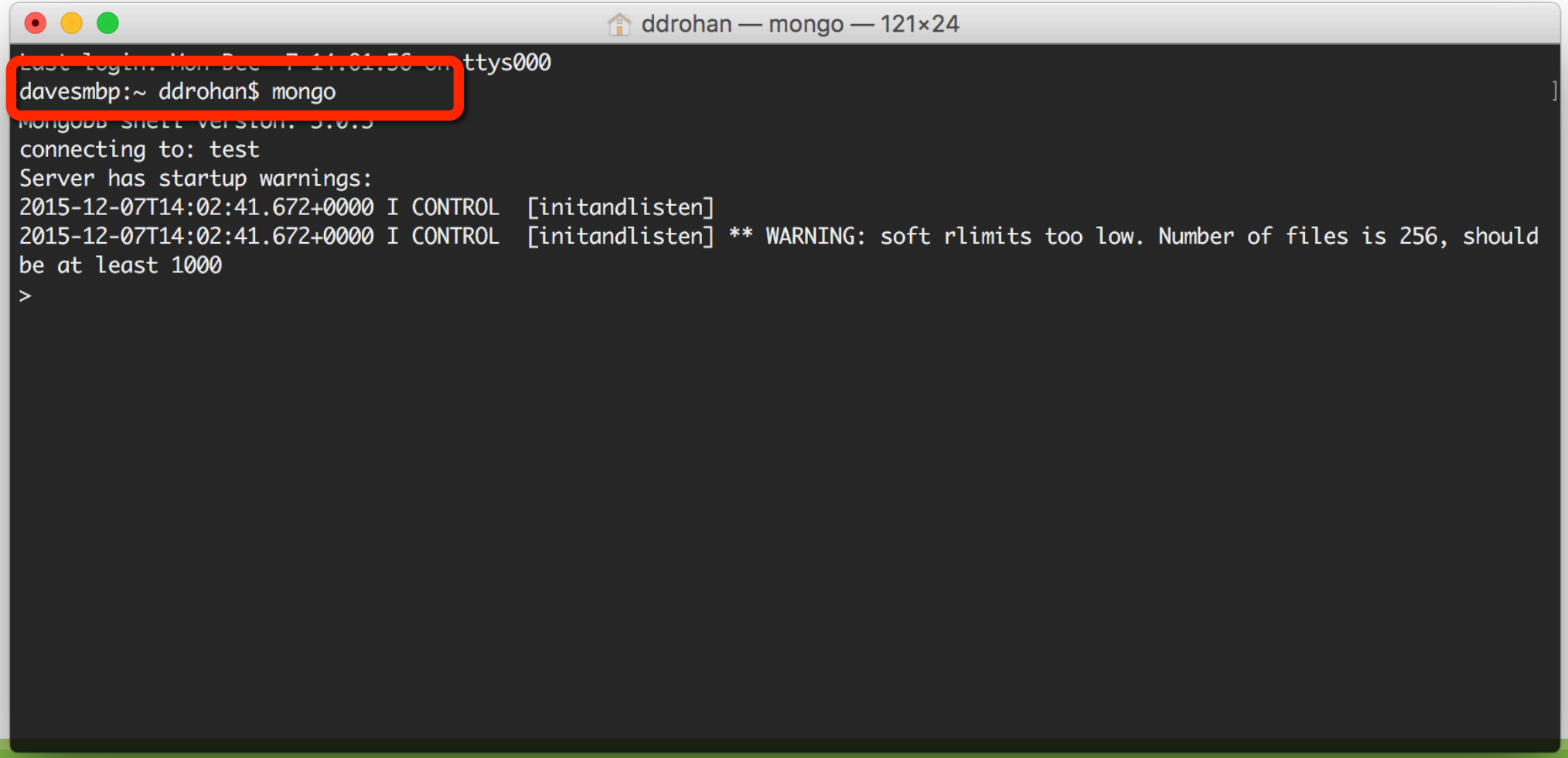
Launch the MongoDB Server



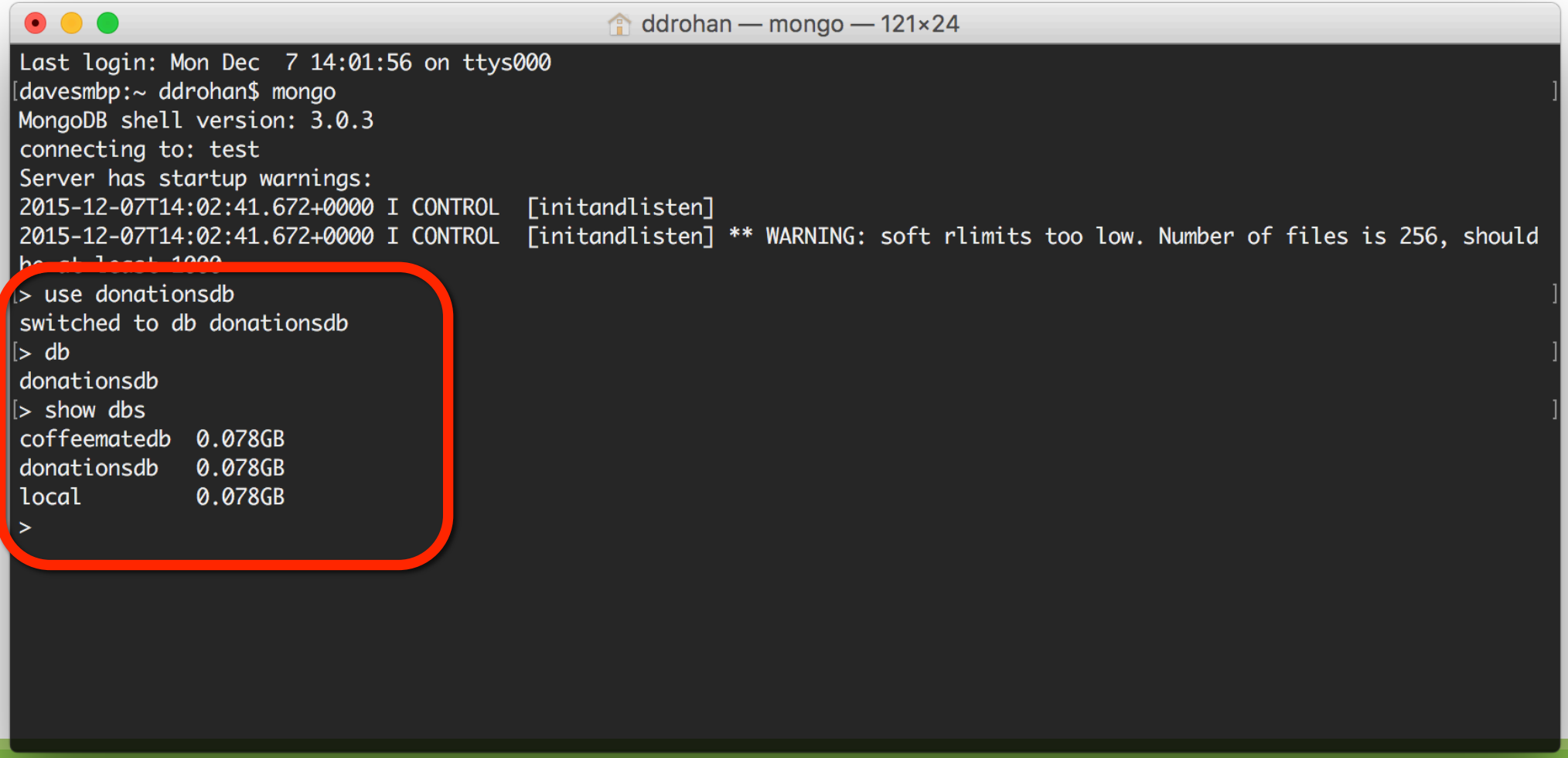
mongoose

mongoDB

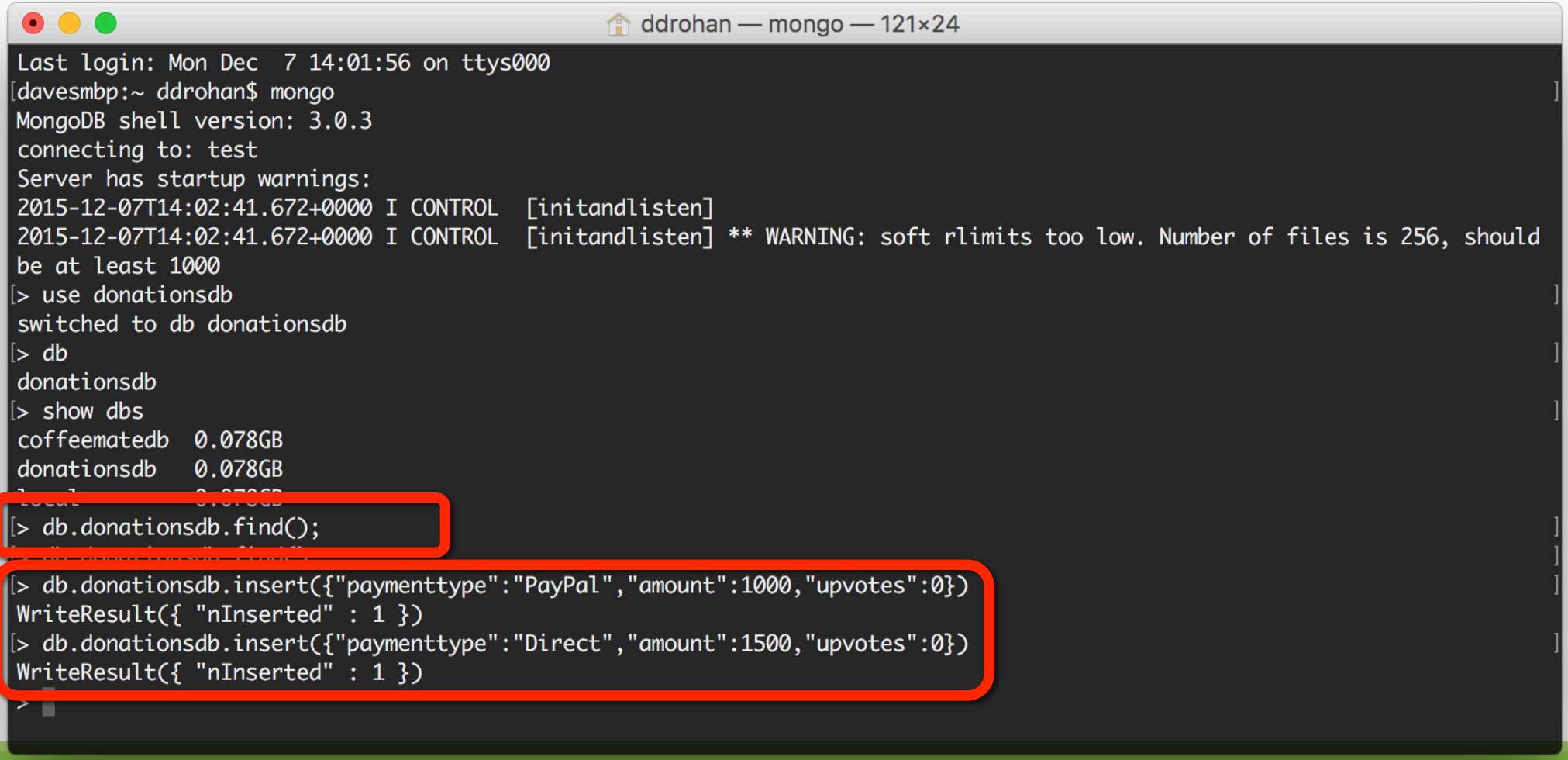
Launch the MongoDB Client



Create/Switch to chosen Database



Search / Insert Records



mongoose

mongoDB

Find All Records

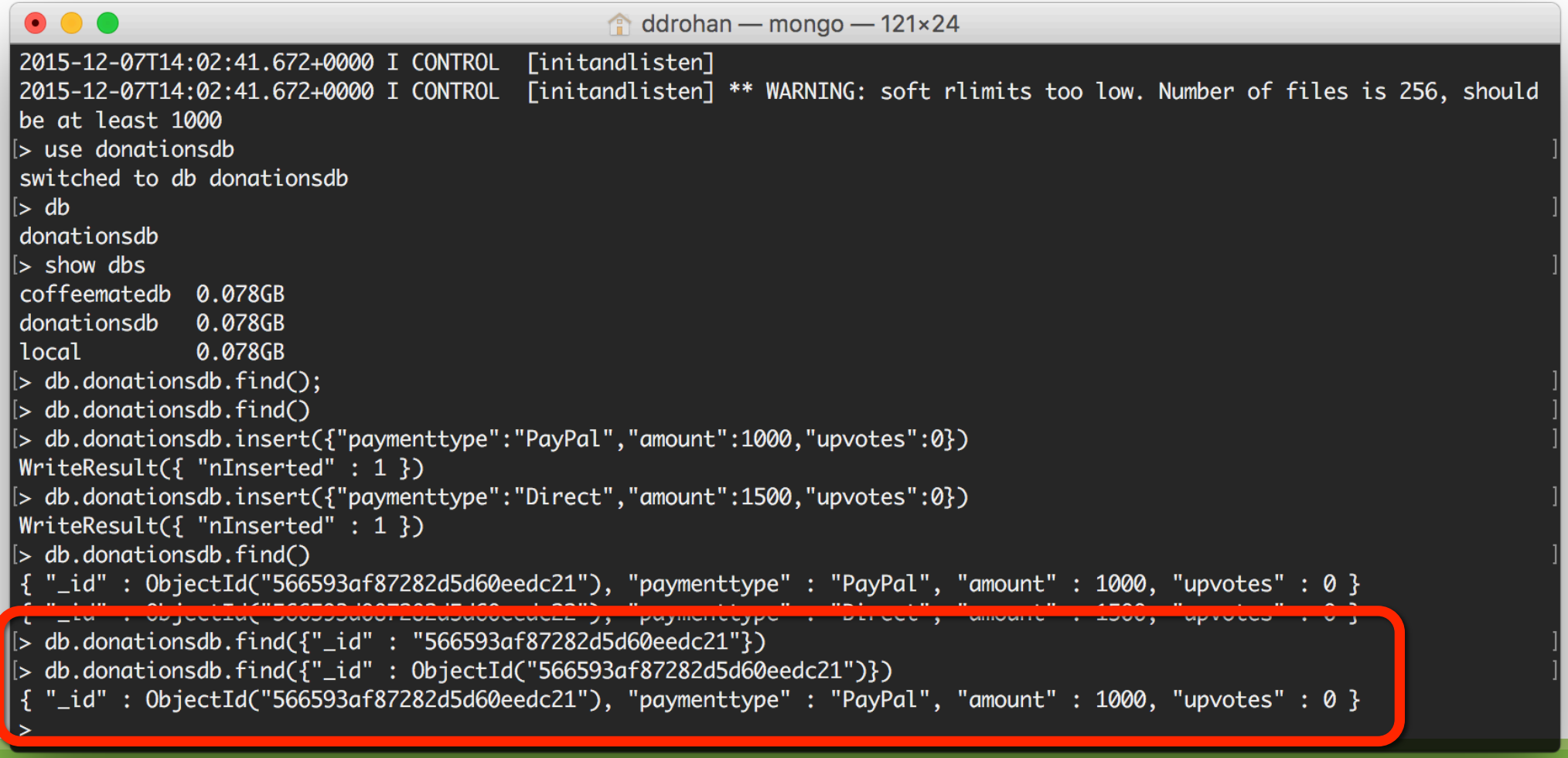
ddrohan — mongo — 121×24

```
MongoDB shell version: 3.0.3
connecting to: test
Server has startup warnings:
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should
be at least 1000
> use donationsdb
switched to db donationsdb
> db
donationsdb
> show dbs
coffeematedb  0.078GB
donationsdb   0.078GB
local         0.078GB
> db.donationsdb.find();
> db.donationsdb.find()
> db.donationsdb.insert({"paymenttype":"PayPal","amount":1000,"upvotes":0})
WriteResult({ "nInserted" : 1 })
> db.donationsdb.insert({"paymenttype":"Direct","amount":1500,"upvotes":0})
WriteResult({ "nInserted" : 1 })
> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
>
```

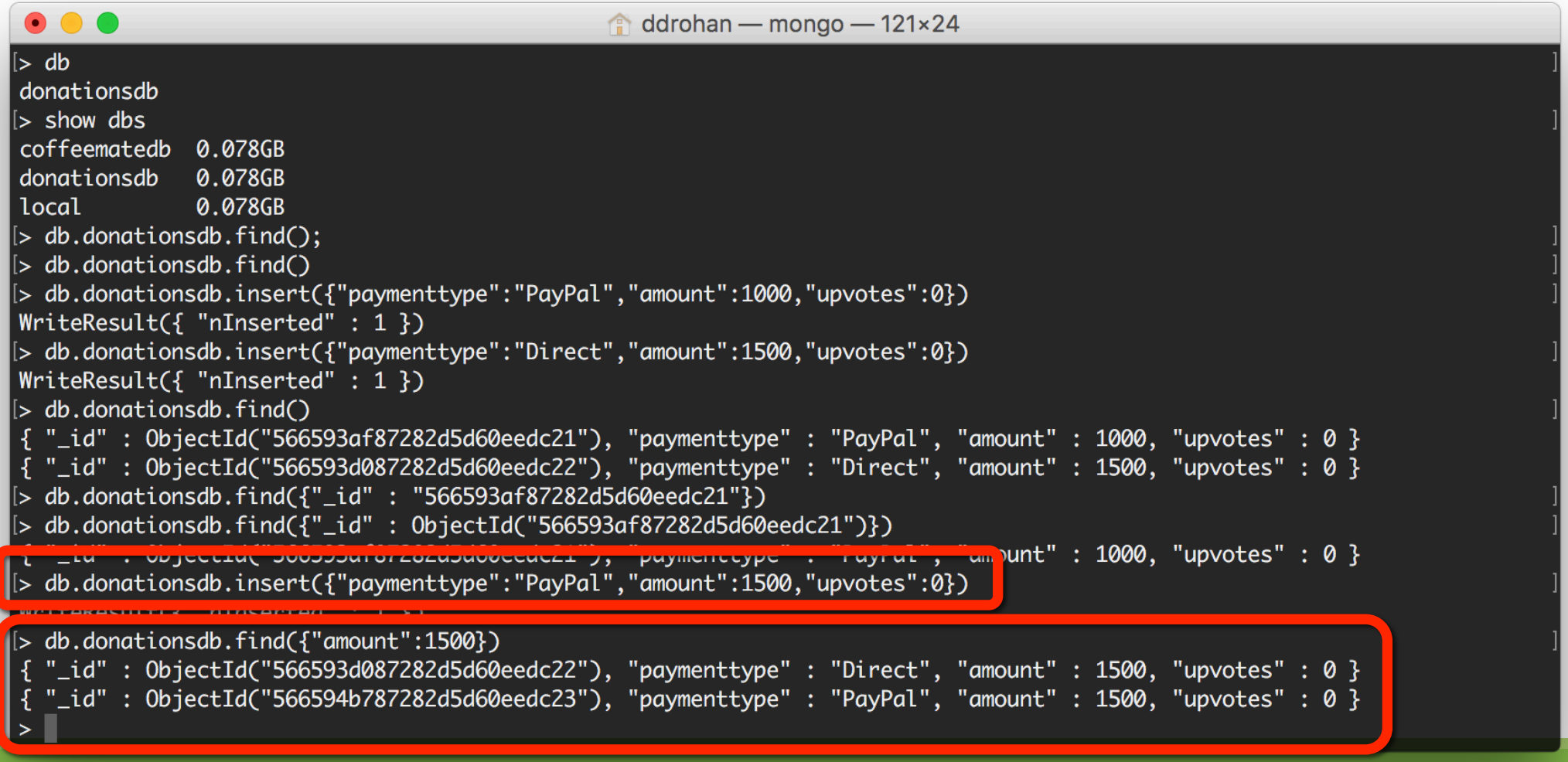
mongoose

mongoDB

Find Records (with criteria)



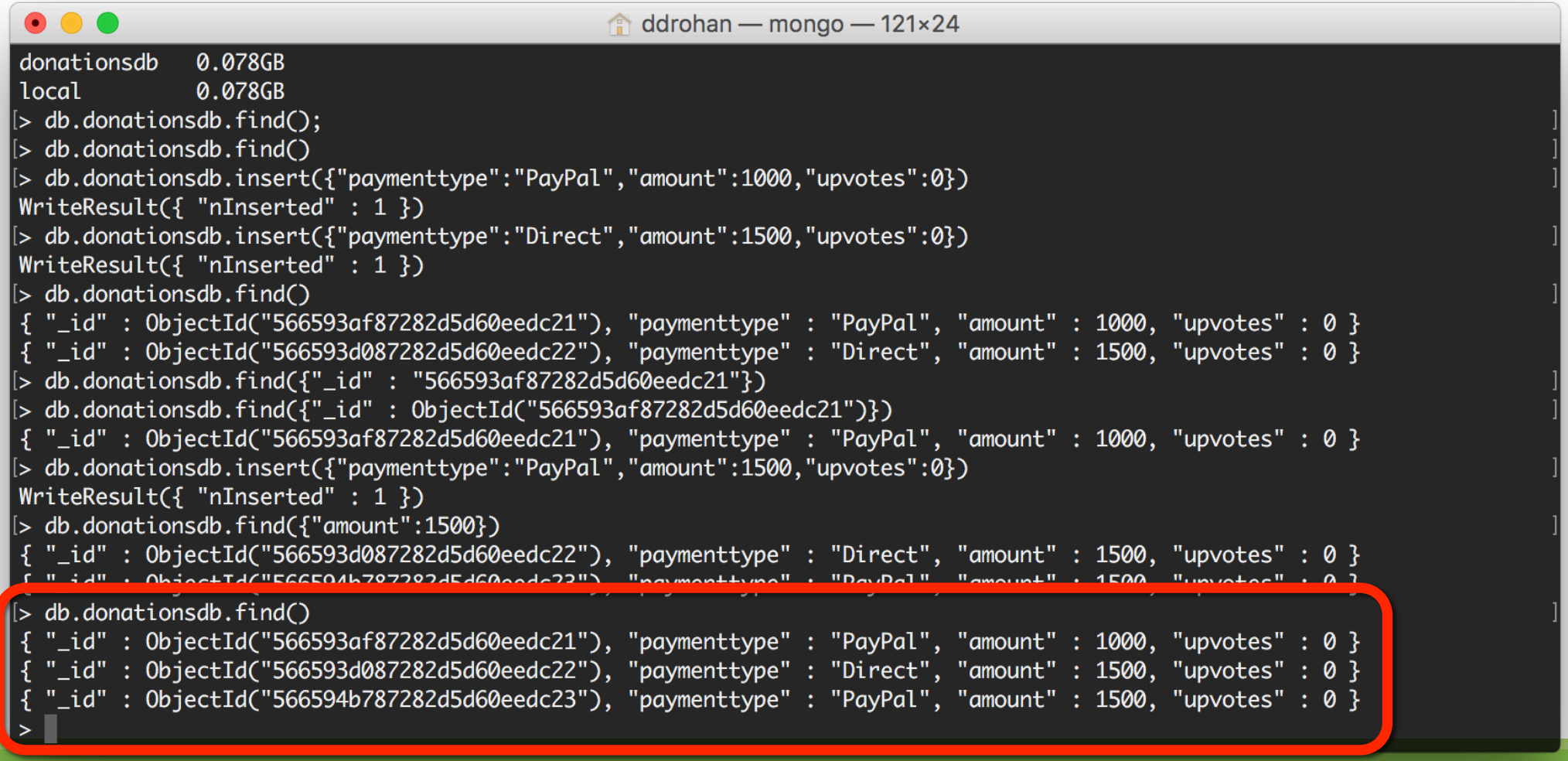
Insert / Find Records (with criteria)



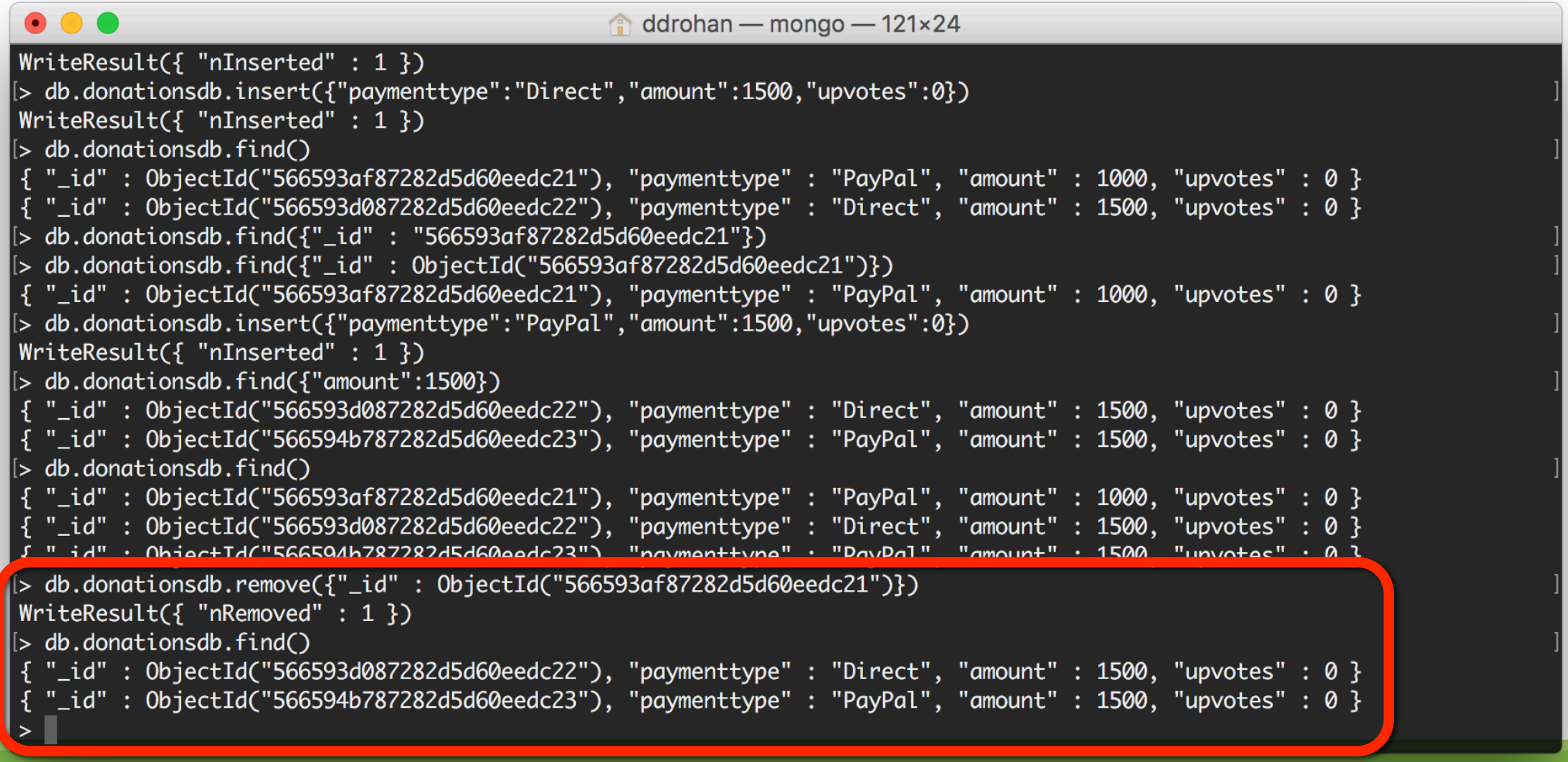
mongoose

mongoDB

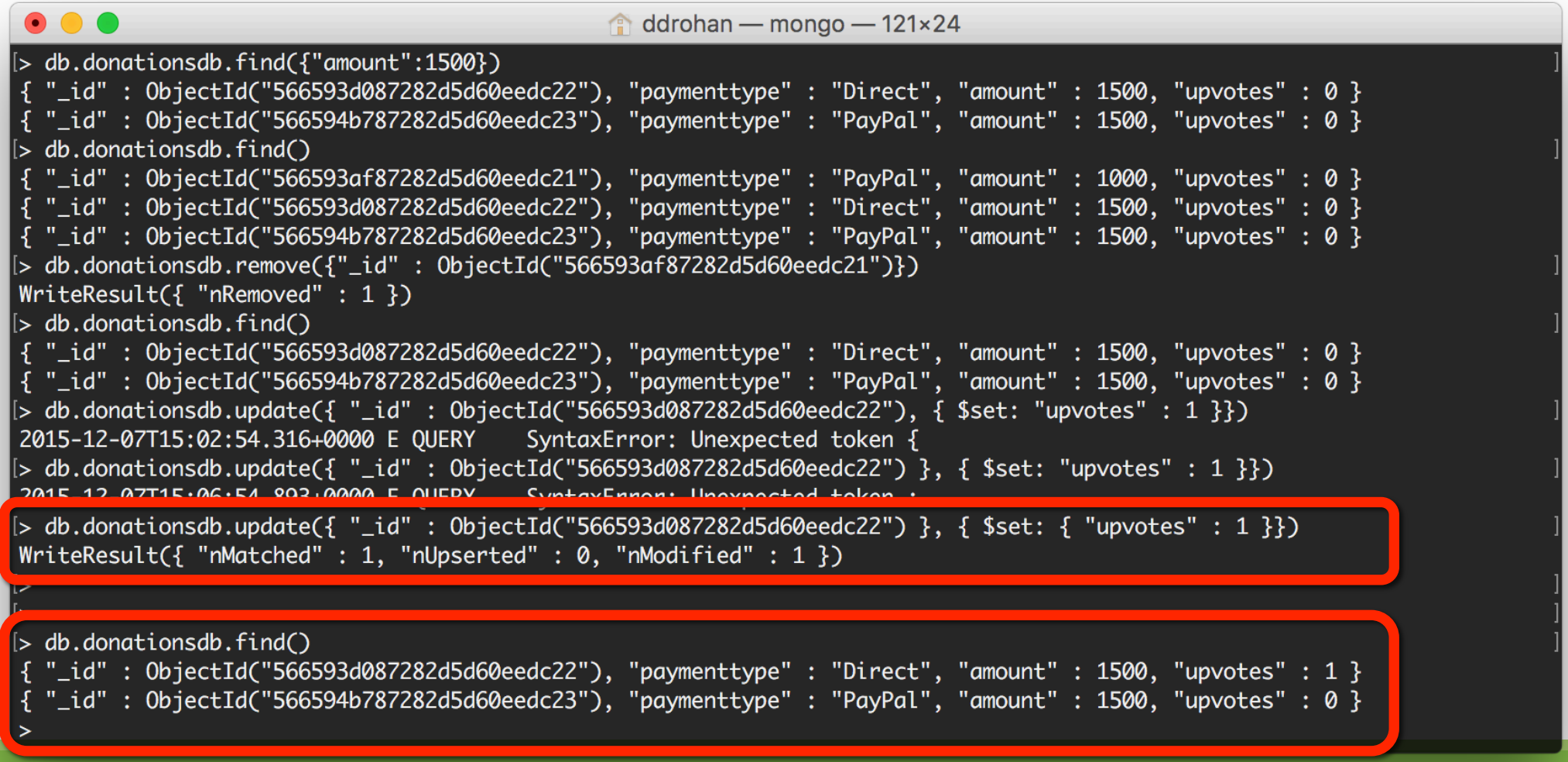
Find all Records



Delete / Remove a Record



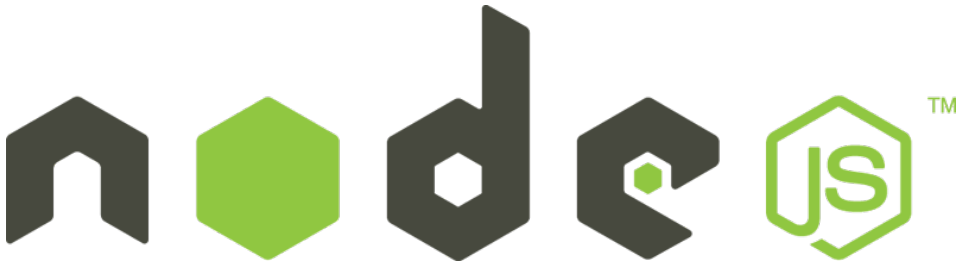
Update a Record



{ name: mongo, type: DB }

MongoDB & Node.js

OBJECT-DOCUMENT MODEL



MongoDB Queries

The MongoDB **module** supports all kinds of queries over the data

- Creating new documents
  - And adding records
- Editing existing documents
  - And their records
- Removing documents and records
- Querying whole documents or parts of them

Using MongoDB with Node.js

Once installed, MongoDB must be started before it can be used

- Go to installation folder and run mongod

```
$ cd path/to/mondodb/installation/folder
$ mongod
```

- Or add mongod.exe to the PATH

When run, MongoDB can be used from **Node.js**

Using MongoDB with Node.js

Download MongoDB from the official web site:

- https://www.mongodb.org/downloads
- Installers for all major platforms

When installed, MongoDB needs a driver to be usable with a specific platform

- One to use with Node.js, another to use with .NET, etc…

Installing MongoDB dependency module for **Node.js:**

```
$ npm install mongodb -g
```

MongoDB Module & Node.js

- The 'mongodb' module is required

```
var mongodb = require('mongodb');
```

- Create a server to host the database

```
var server = new mongodb.Server('localhost', 27017);
```

- Create mongodb client that connects to the server

```
var mongoClient = new mongodb.MongoClient(server);
```

- Open connection to the mongodb server

```
mongoClient.open(function(err, client){
  var db = client.db('DATABASE_NAME');
  //queries over the db
});
```

mongoose

elegant mongodb object modeling for node.js

Mongoose Overview

OBJECT-DOCUMENT MODEL MODULE FOR NODE.JS

Mongoose Overview

Mongoose is a object-document model module in Node.js for MongoDB

- Wraps the functionality of the native MongoDB driver
- Exposes models to control the records in a doc
- Supports validation on save
- Extends the native queries

Installing Mongoose

Run the following from the CMD/Terminal

```
$ npm install mongoose
```

Load the Module

```
var mongoose = require('mongoose');
```

Connect to the Database

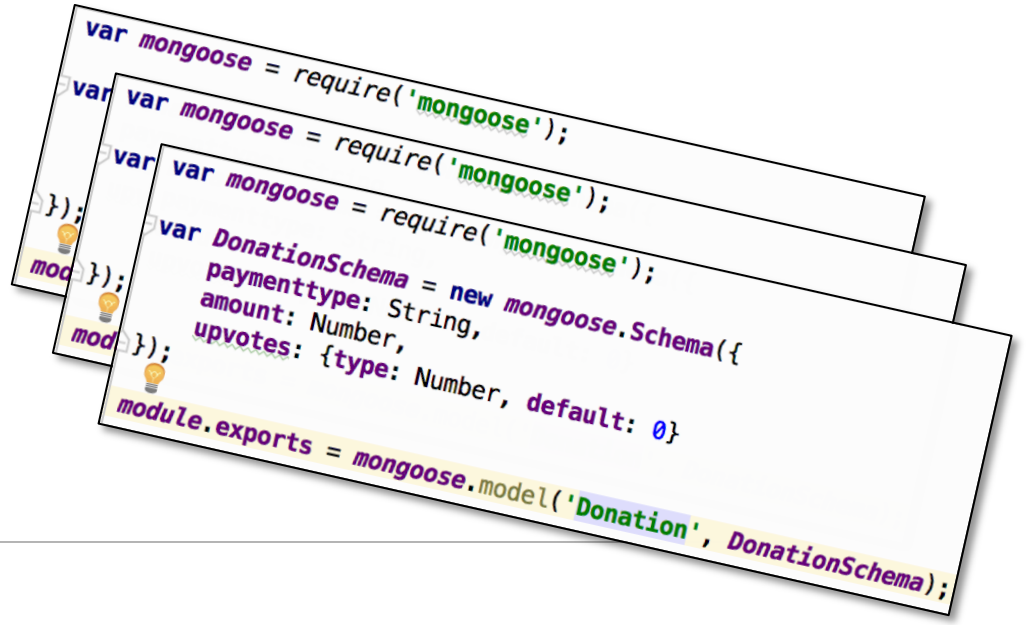
```
mongoose.connect(mongoDbPath);
```

Create Models and persist data

```
var Unit = mongoose.model('Unit', { type: String} );
new Unit({type: 'warrior'}).save(callback); //create
Unit.find({type: 'warrior'}).exec(callback); //fetch
```

mongoose

elegant mongodb object modeling for node.js



Mongoose Models

OBJECT-DOCUMENT MODEL SCHEMA

Mongoose Models

- Mongoose supports Models
  - i.e. fixed types of documents
- Used like object Constructors
  - Needs a ‘mongoose.Schema’

```
var mongoose = require('mongoose');

var DonationSchema = new mongoose.Schema({
    paymenttype: String,
    amount: Number,
    upvotes: {type: Number, default: 0}
});

module.exports = mongoose.model('Donation', DonationSchema);
```

Mongoose Models

- Each of the properties must have a type
  - Types can be Number, String, Boolean, array, object
  - Even nested objects

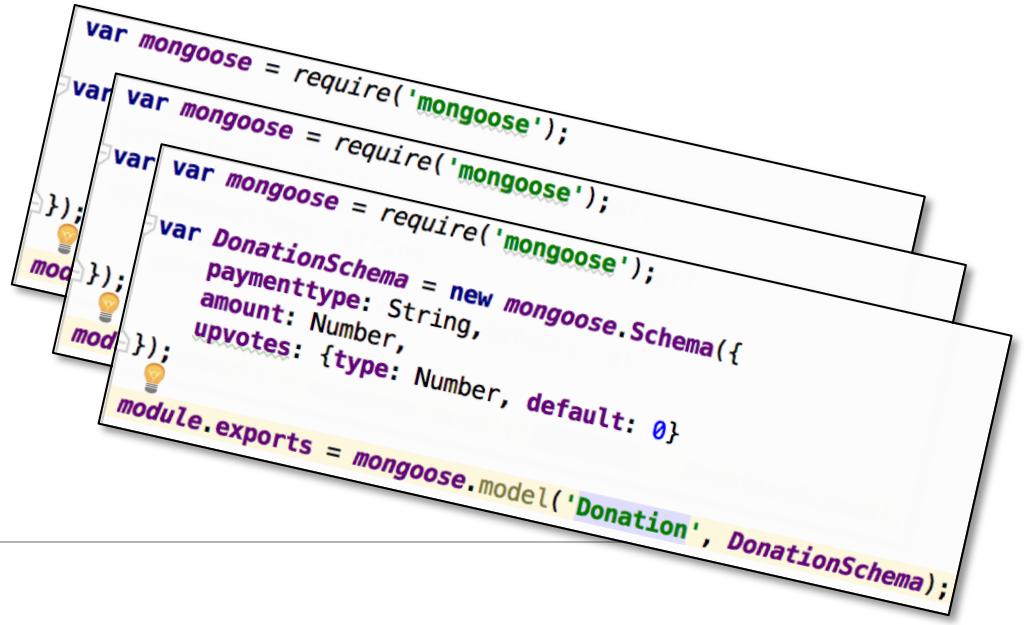
```
var mongoose = require('mongoose');

var DonationSchema = new mongoose.Schema({
    paymenttype: String,
    amount: Number,
    upvotes: {type: Number, default: 0}
});

module.exports = mongoose.model('Donation', DonationSchema);
```

CRUD with Mongoose

OBJECT-DOCUMENT MODEL SCHEMA

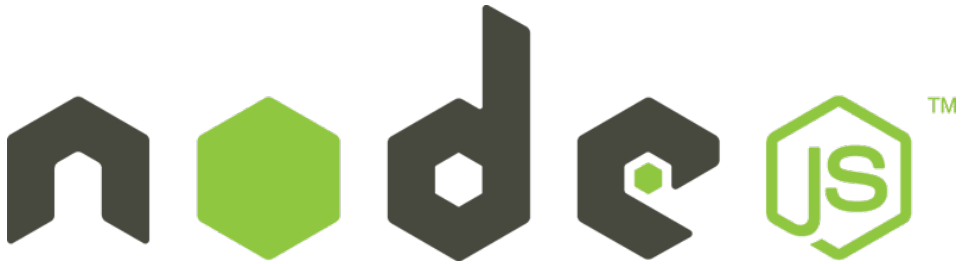
mongoose

elegant mongodb object modeling for node.js



CRUD with Mongoose

- Mongoose supports all the CRUD operations:
  - Create –> modelObj.save(callback)
  - Read –> Model.find().exec(callback)
  - Update –> modelObj.update(props, callback)
    –> Model.update(condition, props, callback)
  - Remove –> modelObj.remove(callback)
    –> Model.remove(condition, props, callback)



{ name: mongo, type: DB }

Donationweb

CASE STUDY EXAMPLE

mongoose

elegant mongodb object modeling for node.js

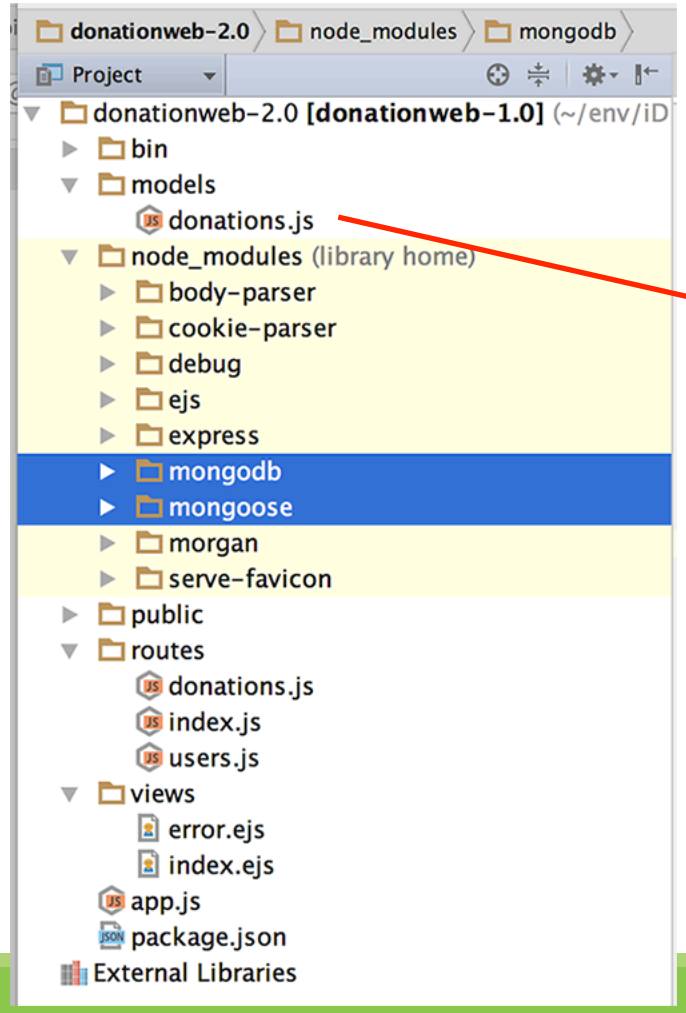
Donation: Resource, URIs & Methods

Resource	URI (structure)	HTTP Request
List of Donations	/donations	GET
Get a Single Donation	/donations/{id}	GET
Upvote a Donation	/donations/{id}/votes	PUT
Delete a Donation	/donations/{id}	DELETE
Update a Donation	/donations/{id}	PUT
Add a Donation	/donations/{id}	POST

{…} = variable value; changeable by user/application to refer to specific resource

We'll look at this Use Case as an example…

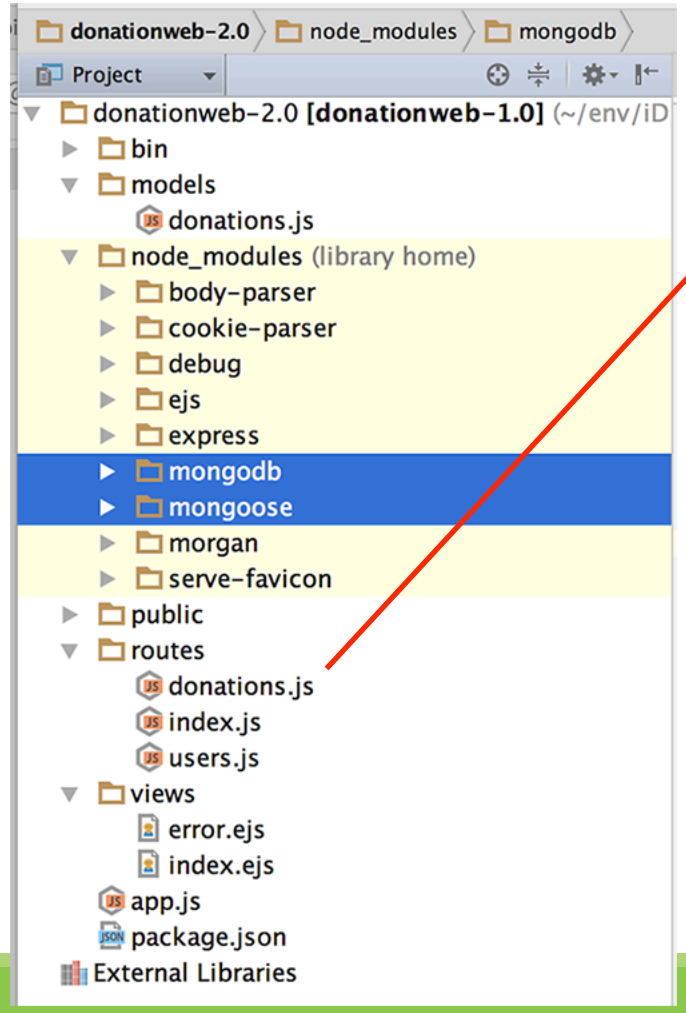
Creating the Model – *Server Side*



models/donations.js × donations.js ×

```
1  var mongoose = require('mongoose');
2
3  var DonationSchema = new mongoose.Schema({
4      paymenttype: String,
5      amount: Number,
6      upvotes: {type: Number, default: 0}
7  });
8
9  module.exports = mongoose.model('Donation', DonationSchema);
```

Creating the Routes (1) – *Server Side*

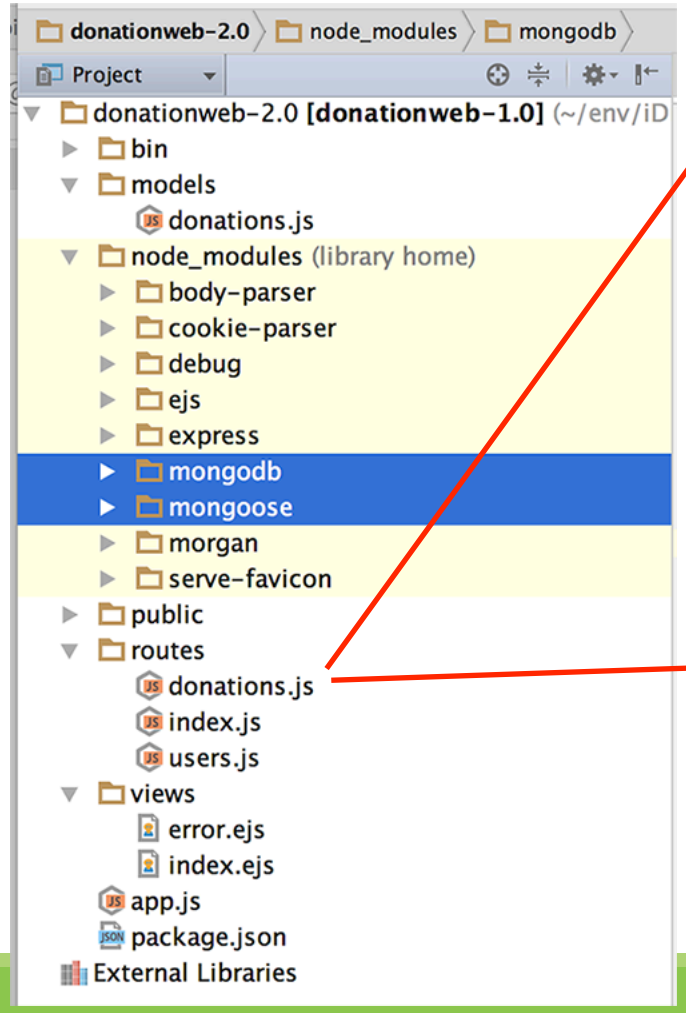


routes/donations.js × models/donations.js ×

```
1   var Donation = require('../models/donations');
2   var express = require('express');
3   var router = express.Router();
4   var mongoose = require('mongoose');
5
6
7   mongoose.connect('mongodb://localhost:27017/donationsdb');
8
9   var db = mongoose.connection;
10
11  db.on('error', function (err) {
12      console.log('connection error', err);
13  });
14  db.once('open', function () {
15      console.log('connected to database');
16  });
17
18  router.findAll = function(req, res) {...}
27
28  router.findOne = function(req, res) {...}
38
39  router.addDonation = function(req, res) {...}
56
57  router.deleteDonation = function(req, res) {...}
66
67  router.incrementUpvotes = function(req, res) {...}
83
84  module.exports = router;
```

N.B. on 'imports'

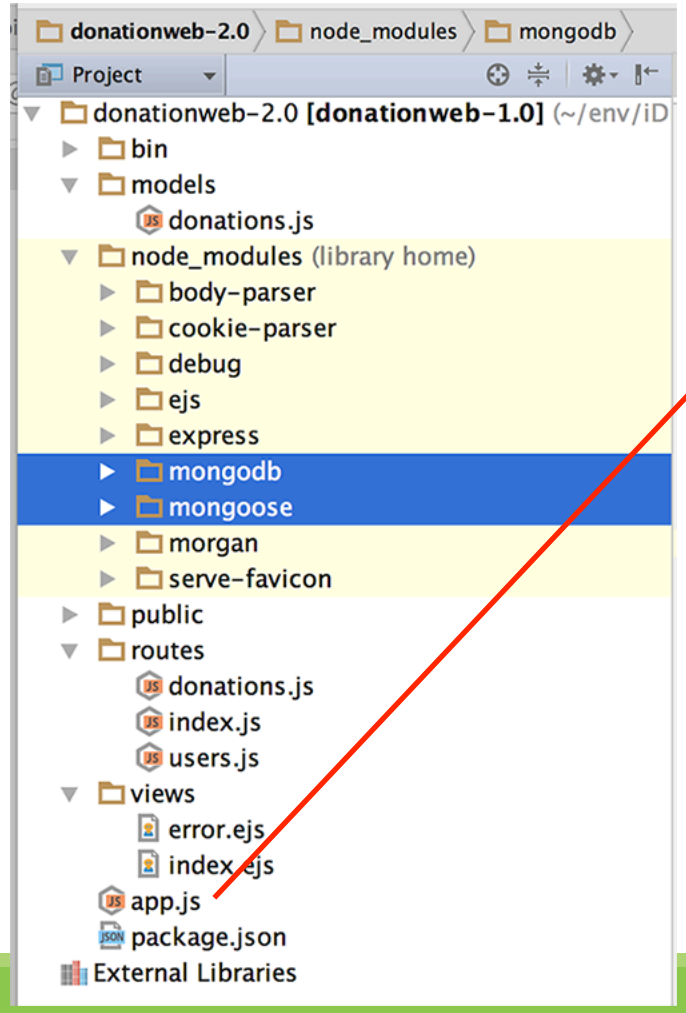
Creating the Routes (2) – *Server Side*




```
router.addDonation = function(req, res) {

    var donation = new Donation();

    donation.paymenttype = req.body.paymenttype;
    donation.amount = req.body.amount;

    //console.log('Adding donation: ' + JSON.stringify(donation));

    // Save the donation and check for errors
    donation.save(function(err) {
        if (err)
            res.send(err);

        res.json({ message: 'Donation Added!', data: donation });
    });
}
```

```
router.findAll = function(req, res) {
    // Use the Donation model to find all donations
    Donation.find(function(err, donations) {
        if (err)
            res.send(err);
        else
            res.json(donations);
    });
}
```

Creating the Routes (3) – *Server Side*



app.js

```
1   var express = require('express');
2   var path = require('path');
3   var favicon = require('serve-favicon');
4   var logger = require('morgan');
5   var cookieParser = require('cookie-parser');
6   var bodyParser = require('body-parser');
7
8   var routes = require('./routes/index');
9   var users = require('./routes/users');
10  var donations = require('./routes/donations.js');
11
12  var app = express();
13
```

```
28
29  //Our Custom Routes
30  app.get('/donations', donations.findAll);
31  app.get('/donations/:id', donations.findOne);
32  app.post('/donations', donations.addDonation);
33  app.put('/donations/:id/votes', donations.incrementUpvotes);
34  app.delete('/donations/:id', donations.deleteDonation);
35
```

```
68
69
70  module.exports = app;
71
```

License

This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

Attribution: this work may contain portions from

- "Databases" course by Telerik Academy under CC-BY-NC-SA license and

mongoose

Great Resources

Official Tutorial – https://nodejs.org/documentation/tutorials/

Official API – https://nodejs.org/api/

Developer Guide – https://nodejs.org/documentation

Video Tutorials – http://nodetuts.com

Video Introduction – https://www.youtube.com/watch?v=FqMIyTH9wSg

YouTube Channel – https://www.youtube.com/channel/UCvhIsEIBIfWSn_Fod8FuuGg

Articles, explanations, tutorials – https://nodejs.org/community/

Questions?