THE UNIVERSITY OF WESTERN AUSTRALIA

Topic 12: Connecting Express and Mongo

CITS3403 Agile Web Development

Getting MEAN with Mongo, Express, Angular and Node, Chapter 5 and 6

Semester 1, 2018

Node and Mongo

THE UNIVERSITY OF WESTERN AUSTRALIA

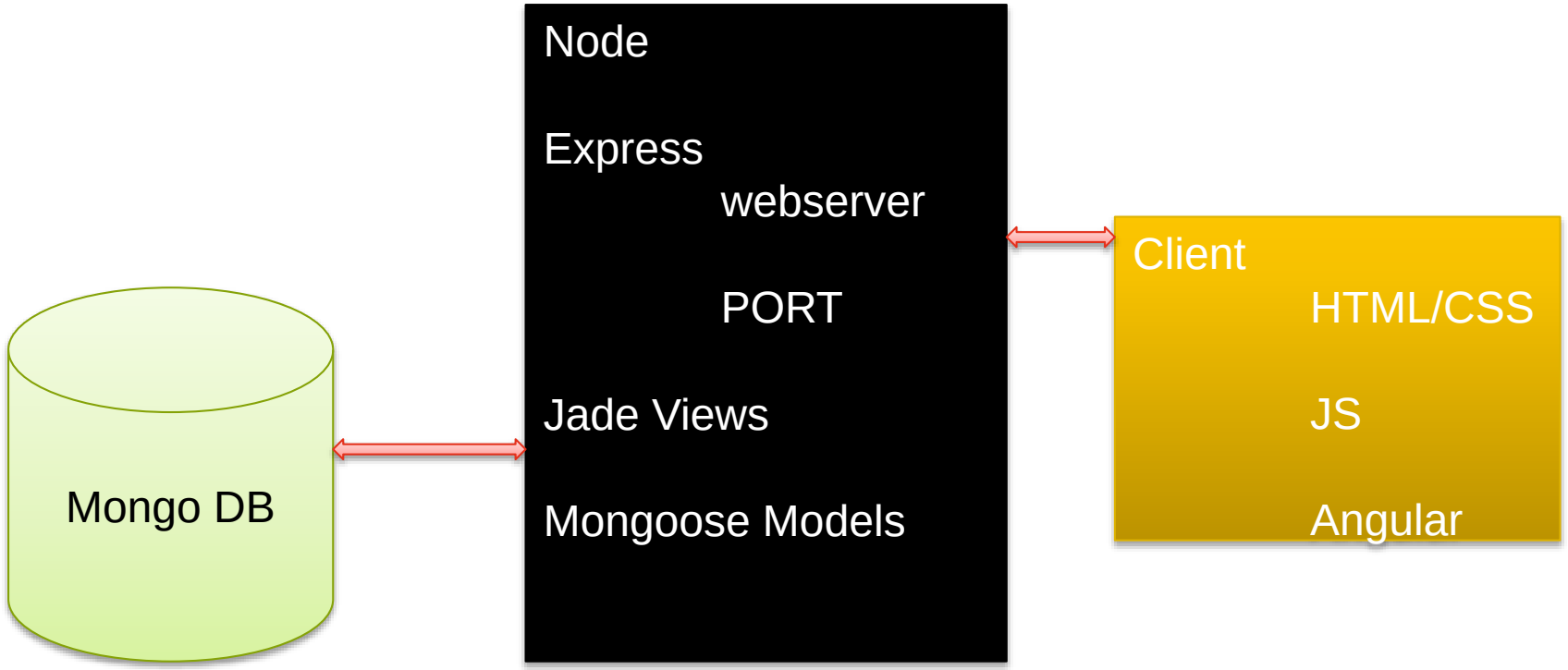
- There are several ways to connect an express application to a mongo database.
- From you can install mongodb from npm which gives a set of functions to access a mongo database directly.

```
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();
});
```

It is important to use callbacks correctly here, so you don't close a connection that is being used.

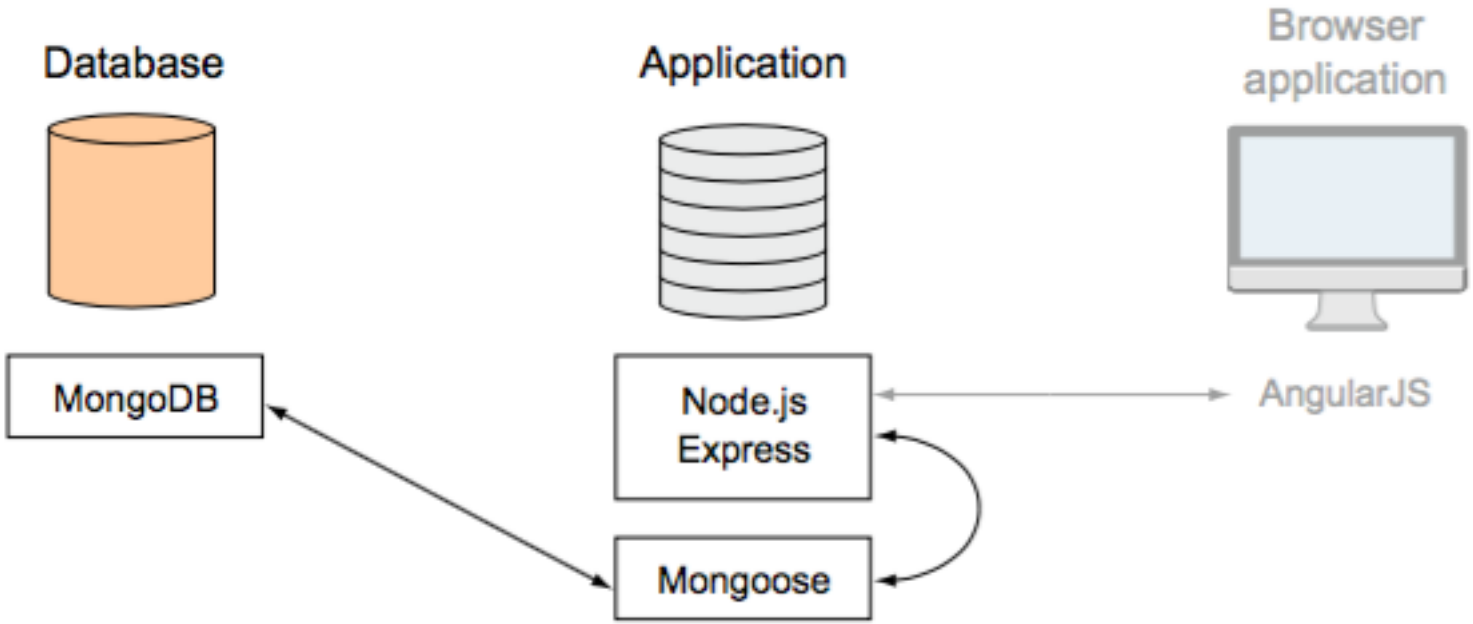
Mongo and Express

Mongo is a document database, with no enforced schema, so we need a *model* to enforce data integrity. For this we will use mongoose.



Mongoose

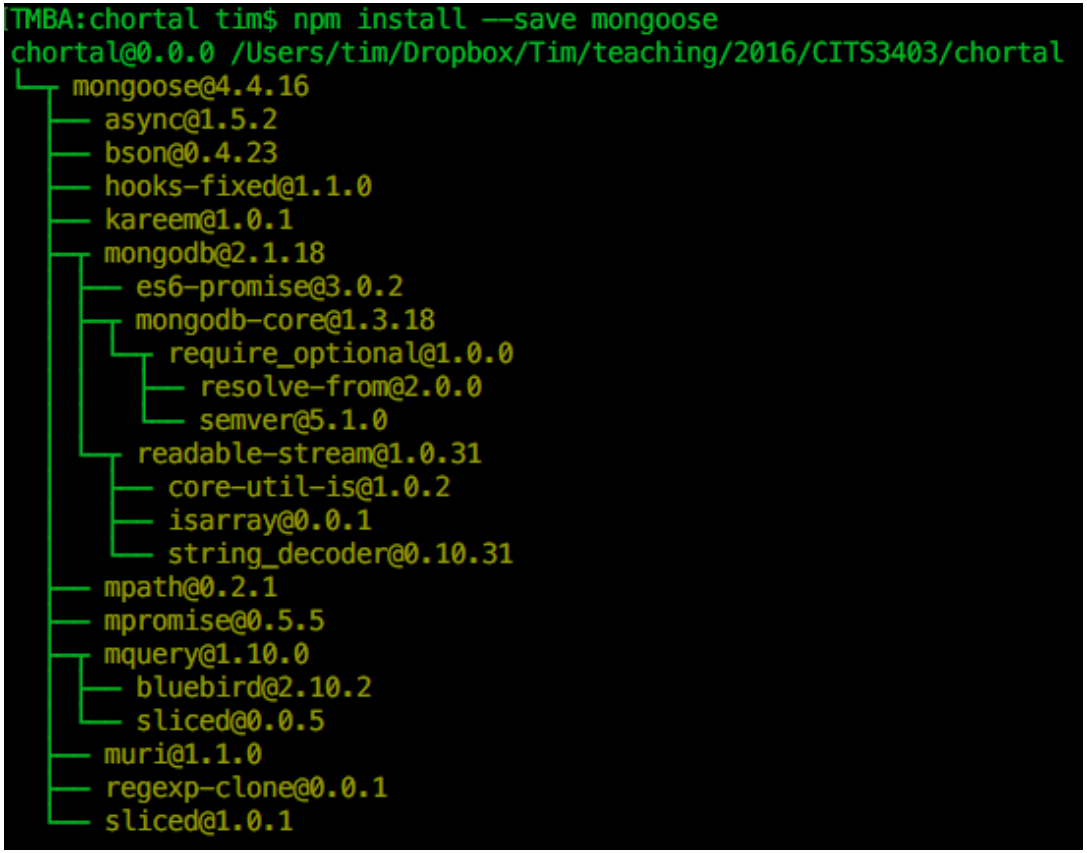
- Mongoose wraps the functionality of the mongodb database in a model, making it easy to work with.
- Mongoose allows us to define models and schemas, and saves us writing validation code.



Installing Mongoose

- Add Mongoose to the application using npm.
- This makes mongoose available throughout the express app.
- We need to

1. connect to a database
2. build schemas for the models
3. write and read data from mongo.



Connecting to the database

- Connecting to a database takes time, so mongoose tries to reuse connection where possible.
- We will setup the connection in the *models* directory.
- The connection will be in a file `db.js` that we will then be able to import into the other models.
- We can *require* this file in the `app.js` file so the connection is made as soon as the app starts.

```
1  var mongoose = require('mongoose');
2
3  var dbURI = 'mongodb://localhost/chortal';
4  mongoose.connect(dbURI);
5
6  mongoose.connection.on('connected', function () {
7      console.log('Mongoose connected to ' + dbURI);
8  });
9  mongoose.connection.on('error',function (err) {
10     console.log('Mongoose connection error: ' + err);
11 });
12 mongoose.connection.on('disconnected', function () {
13     console.log('Mongoose disconnected');
14 });
15
16 var gracefulShutdown = function (msg, callback) {
17     mongoose.connection.close(function () {
18           console.log('Mongoose disconnected through ' + msg);
19               callback();
20                 });
21 };
```

Modeling the data

THE UNIVERSITY OF WESTERN AUSTRALIA

- Even though Mongo does not require structured data, good design still requires data schemas.
- If we consider the users of our system, they should have:
  - a name (string, must be there),
  - an age (optional, an integer),
  - a list of chores (essential, but could be empty),
  - maybe a picture…
- Everytime we read and write data to the database we would like to enforce these constraints.

Mongoose Models

- Mongoose is an Object Document Modeler (ODM, not ORM). It provides an interface to the database from the application.

- In MongoDB each entry in a database is called a *document*.
- In MongoDB a collection of documents is called a *collection.*
- In Mongoose the definition of a document is called a *schema*.
- Each individual data entity defined in a schema is called a *path*.

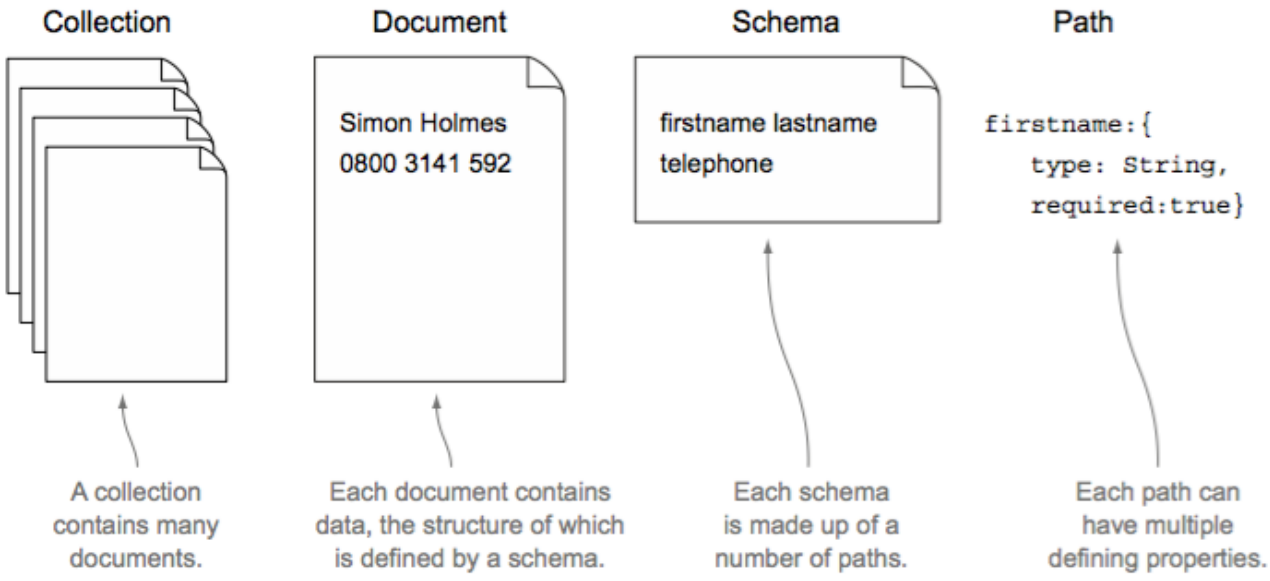


Figure 5.7 Relationships among collections, documents, schemas, and paths in MongoDB and Mongoose, using a business card metaphor

Mongoose Schemas

THE UNIVERSITY OF WESTERN AUSTRALIA

- A mongoose schema allows you to define the fields of your model and specify the constraints on those fields.
  - Like Mongo Documents they can be nested.
  - A sample object in our application may look like this.
  - We can use this sketch to define a schema.

```
Minion example:
{name: 'Minion 1',
 age: 7,
 chores: [
   {name: 'Kill turret',
    desc: 'Turret must be dropped from a great height',
    days: [0,1,2,3,4,5],
    stars: 1,
    completed: [2016-06-10,
                2016-06-11]
   },
   {name: 'Resurrect GLaDOS',
    desc: 'restart GLaDOS for the cakes',
    days: [6],
    stars: 5,
    completed: [2016-06-05]
   }
 ],
 stars: 7,
 picture: <binary data>
}
```

Schema specification

- A schema specification gives the data type and constraints of each field in the system.
- The following file, `minion_model.js` can specify the schema:
  - If we require this file at the *end* of `db.js` then the model will be bought into the whole application.

```
1  var mongoose = require('mongoose');
2
3  var choreSchema = new mongoose.Schema(
4      {name:{type:String, require:true},
5        desc:{type:String, required:false},
6        days:{type:[Number], required:true, 'default':[6], min:0, max:6},
7              //days assume 0 = Sunday,...,6=saturday
8        stars:{type:Number, required:true, 'default':1},
9        completed:{type:[Date], required: true, 'default':[]}
10     });
11
12 var minionSchema = new mongoose.Schema(
13     {name:{type: String,required:true},
14       age: {type: Number, min:0, max:99},
15       chores: {type:[choreSchema], required:true},
16       stars: {type:Number, required:true, 'default':0},
17       picture: {type: Buffer, required:false}});
18
19 mongoose.model('Minion', minionSchema);
20
```

Compiling Schemas into Models

THE UNIVERSITY OF WESTERN AUSTRALIA

- To build a model from a schema, mongoose needs to know the name of the model, the schema to use, and the collection in the database to use.
- `mongoose.model('Minion', minionSchema, minions)`
- You can place this line after the schema definition in `minion_model.js`
- You are then able to create new models of Minions throughout the app.

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var Cat = mongoose.model('Cat', { name: String });

var kitty = new Cat({ name: 'Zildjian' });
kitty.save(function (err) {
  if (err) {
    console.log(err);
  } else {
    console.log('meow');
  }
});
```