MongoDB – part 2

Getting into js

Mongo and friend

- Mongoose is a Mongo Based Object Document Modeling DB
- Its based on MongoDb
- We will be using it, as it is easier to develop in, and has better docs.
- Docs: http://mongoosejs.com/index.html

Local Testing

- כאשר עובדים עם מונגו בצורה לוקאלית, צריך לזכור תמיד להפעיל את מונגו בCMD נוסף
- נעשה זאת עם הפקודה mongod
- לאחר מכן נוכל להתחבר לשירות זה בתוך הקוד
- אם נשכח לעשות את זה, שום דבר לא באמת ישתנה, ולרוב השרת לעולם לא יסיים את פעולת הכתיבה

First steps

- First we need to install it – npm install mongoose
- Then we need to require it in the code – var mongoose = require(‘mongoose’);
- We connect to mongo backend - mongoose.connect('mongodb://localhost/test');

http://mongoosejs.com/docs/index.html

הגדרות בסיסיות

- Model - זהו האובייקט שמייצג את האוסף במונגו
- Schema - יש לנו יכולת להגדיר מבנה של מסמכים באוסף

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

Model.find(conditions, [projection], [options], [callback])

- המקבילה לחיפוש במונגו שראינו.
- מקבלת אובייקט תנאים - conditions
- מקבלת אובייקט החזרה - projection
- מקבלת אובייקט אפשרויות נוספות
- מקבלת פונקציית callback להתמודדות עם הצלחה או כשלון

Find methods - http://mongoosejs.com/docs/api.html#model_Model.find

שמירה

```
var Tank = mongoose.model('Tank', yourSchema);//create model
var small = new Tank({ size: 'small' });//init the model
//save to db
small.save(function (err) {
        if (err) return handleError(err); // saved!
})
```

יצירת פונקציות למסמכים

- ניתן להגדיר פונקציות למסמכים
- כך נוכל להגדיר בעצם טיפול בבעיות או חיפושים מיוחדים לכל סוג של נתון שלנו

```
// define a schema
var animalSchema = new Schema({ name: String, type: String });
 // assign a function to the "methods" object of our animalSchema
animalSchema.methods.findSimilarTypes = function (cb) {
        return this.model('Animal').find({ type: this.type }, cb);
 }
```

- פה יצרנו פונקציה בשם findSimilarTypes עבור המודל Animal

built-in instance methods

הגדרת פונקציות סטטיות למודלים

- נוכל להגדיר גם פונקציה למודל

```
// assign a function to the "statics" object of our animalSchema
animalSchema.statics.findByName = function (name, cb) {
  return this.find({ name: new RegExp(name, 'i') }, cb);
}

var Animal = mongoose.model('Animal', animalSchema);
Animal.findByName('fido', function (err, animals) {
  console.log(animals);
});
```

שמירת על אוסף ממוין

- אנחנו יכולים להגדיר סדר הופעה בתוך אוסף כדי להקל על החיפוש
- את ההגדרה אפשר לעשות ברמת הנתון או ברמת הסכימה

```
var animalSchema = new Schema({
  name: String,
  type: String,
  tags: { type: [String], index: true } // field level
});

animalSchema.index({ name: 1, type: -1 }); // schema level
```

הוספת משתנים וירטואליים

- לפעמים אנחנו רוצים להגדיר משתנים דינאמיים
- המשתנים אינם נשמרים במסמך בפועל
- נרצה להגדיר אותם כדי להקל על עצמנו בעבודה
- למשל נרצה להגדיר משתנה המחזיר שם מלא של בנאדם כאשר במסמך יש משתנים של שם פרטי ושם משפחה

הוספת משתנים וירטואליים

```
// define a schema
var personSchema = new Schema({
  name: {
    first: String,
    last: String
  }
});
var Person = mongoose.model('Person', personSchema);

// create a document
var bad = new Person({
    name: { first: 'Walter', last: 'White' }
});
```

~~console.log(bad.name.first + ' ' + bad.name.last); // Walter White~~

```
personSchema.virtual('name.full')
.get(function () {
  return this.name.first + ' ' + this.name.last;
});

console.log('%s is insane', bad.name.full); // Walter White is insane
```

מחיקה

- כמובן שניתן למחוק מאוסף
- מבנה: model.remove([condition:Object], Callback);

```
var Tank = mongoose.model('Tank', yourSchema);

Tank.remove({ size: 'large' }, function (err) {
  if (err) return handleError(err);
  // removed!
});
```

- במקרה הזה מחקנו מTank את כל מה שמכיל size: 'large'