Telerik Academy

MongoDB and Mongoose

Object-document Model

Telerik Software Academy
Learning & Development
http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

MongoDB Native Overview

Using MongoDB

- 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 driver for Node.js:

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

Telerik Academy

Working with MongoDB from Node.js

- Once installed, the MongoDB must be started
  - Go to installation folder and run mongod

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

  - Or add mongod.exe to the PATH
- When run, the MongoDB can be used from Node.js

Telerik Academy

Creating MongoDB Database

- The database is created using 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
});
```

Telerik Academy

Queries over MongoDB with Node.js

- 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

Telerik Academy

Node.js and MongoDB

Live Demo

Telerik Academy

Mongoose Overview

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
```

- In node
  - 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
```

Telerik Academy

Installing and Using Mongoose

Live Demo

Telerik Academy

Mongoose Models

Mongoose Models

- Mongoose supports models
  - i.e. fixed types of documents
    - Used like object constructors
  - Needs a mongoose.Schema

```
var modelSchema = new mongoose.Schema({
  propString: String,
  propNumber: Number,
  propObject: {},
  propArray: [],
  propBool: Boolean
});

var Model = mongoose.model('Model', modelSchema);
```

Mongoose Models (2)

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

```
var modelSchema = new mongoose.Schema({
  propNested: {
    propNestedNumber: Number,
    propDoubleNested: {
      propArr: []
    }
});

var Model = mongoose.model('Model', modelSchema);
```

Telerik Academy

Mongoose Models

Live Demo

Telerik Academy

Mongoose Models with Instance Methods

- Since mongoose models are just JavaScript object constructors they can have methods
  - And these methods can be added to a schema
    - Use a different syntax than plain JS

```
var unitSchema = new mongoose.Schema({…});
unitSchema.methods.move = function(to){
 …
};
```

- And now can be called on a model of type Unit

```
var unit = new Unit({ … } );
unit.move({x: 5, y: 6});
```

Telerik Academy

Mongoose Models with Instance Methods

Live Demo

Telerik Academy

Mongoose Models with Virtual Properties

- Yet, not all properties need to be persisted to the database
  - Mongoose provides a way to create properties, that are accessible on all models, but are not persisted to the database
    - And they have both getters and setters

```
var unitSchema = new mongoose.Schema({…});
game.virtual('escapedTitle').get(function(){ … });
game.virtual('escapedTitle').set(function(title){ … });
```

Telerik Academy

Virtual Properties

Live Demo

Telerik Academy

Property Validation

Property Validation

- With Mongoose developers can define custom validation on their properties
  - i.e. validate records when trying to save

```
var unitSchema = new mongoose.Schema({…});
unitSchema.path('position.x').validate(function(value){
  return value>=0 && value <= maxX;
});
unitSchema.path('position.y').validate(function(value){
  return value>=0 && value <= maxY;
});
```

Telerik Academy

Property Validation

Live Demo

Telerik Academy

CRUD with Mongoose

Telerik Academy

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, cb)
  - Remove –> modelObj.remove(callback)
    –> Model.remove(condition, props, cb)

Telerik Academy

CRUD Operations with Mongoose

Live Demo

Telerik Academy

Mongoose Queries

Mongoose Queries

- Mongoose defines all queries of the native MongoDB driver in a more clear and useful way
  - Instead of:

```
{$or: [{conditionOne: true},
       {conditionTwo: true}]
}
```

  - Do:

```
.where({conditionOne: true}).or({conditionTwo: true})
```

Telerik Academy

Mongoose Queries (2)

- Mongoose supports many queries:
  - For equality/non-equality
  - Selection of some properties
  - Sorting
  - Limit & skip
- All queries are executed over the object returned by Model.find*()
  - Call .exec() at the end to run the query

Telerik Academy

Mongoose Queries

Live Demo

Telerik Academy

Mongoose Models Modules

Mongoose Models

- Having all model definitions in the main module is no good
  - That is the reason Node.js has modules in the first place
  - We can put each model in a different module, and load all models at start

Telerik Academy

Mongoose Models Modules

Live Demo

Telerik Academy

MongoDB and Mongoose

http://academy.telerik.com

Homework

1. Create a modules for Chat, that keep the data into a local MongoDB instance
   - The module should have the following functionality:

```
var chatDb = require('chat-db');
//inserts a new user records into the DB
chatDb.registerUser({user: 'DonchoMinkov', pass: '123456q'});

//inserts a new message record into the DB
//the message has two references to users (from and to)
chatDb.sendMessage({
  from: 'DonchoMinkov',
  to: 'NikolayKostov',
  text: 'Hey, Niki!'
});

//returns an array with all messages between two users
chatDb.getMessages({
  with: 'DonchoMinkov',
  and: 'NikolayKostov
});
```