Full Stack III - Lecture 6.1

Intro to Mongoose

Topics

- Mongoose
- Schema and Model
- CRUD Operations
- Queries

Mongoose

Mongoose

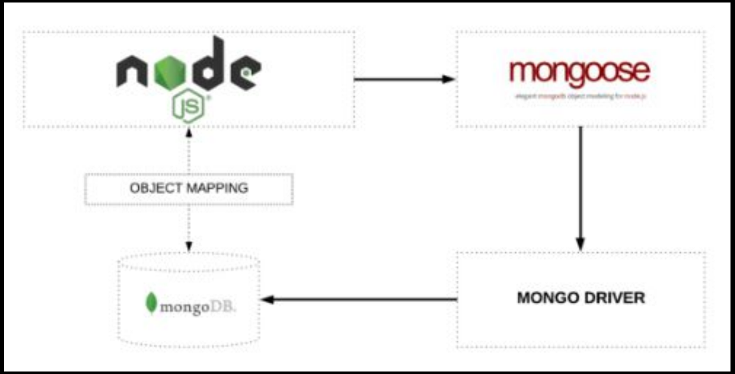
mongoose

elegant mongodb object modeling for node.js

https://mongoosejs.com/

- Alternative middleware to MongoDB Driver
- Mongoose is an object modeling package for Node that essentially works like an ODM *(Object Document Mapper)*
- Mongoose allows us to have access to the MongoDB commands for CRUD simply and easily.

Mongoose cont..



- The Mongoose document is modeled directly from the MongoDB document, thus not an ORM
- Mongoose provides some features that MongoDB does not:
  - Validation
  - Defaults
  - Query builder
  - Pseduo-joins
  - Life-cycle management

Using Mongoose

- Install it via npm

```
$ npm install mongoose --save
```

- Require the package in our project

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

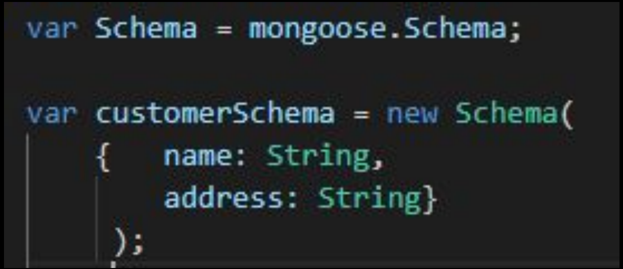
- Connect to the MongoDB

```
mongoose.connect('mongodb://localhost/myappdatabase');
```

Schema & Model

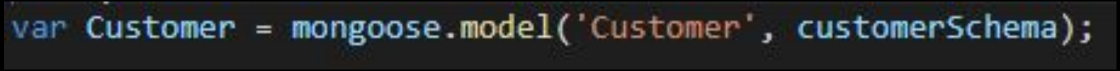
Defining Mongoose Schemas & Models

Everything in Mongoose starts with a Schema. Each schema maps to a MongoDB collection and defines the shape of the documents within that collection.



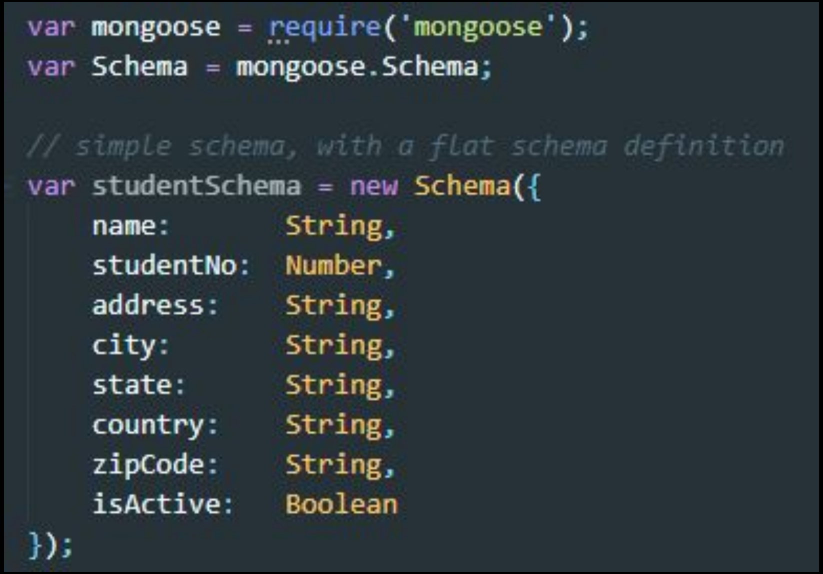
```
var Schema = mongoose.Schema;

var customerSchema = new Schema(
    {   name: String,
        address: String}
    );
```

Creating a model from our schema definition

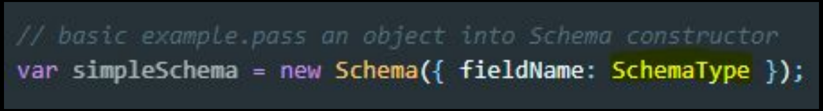


```
var Customer = mongoose.model('Customer', customerSchema);
```

Mongoose Schema



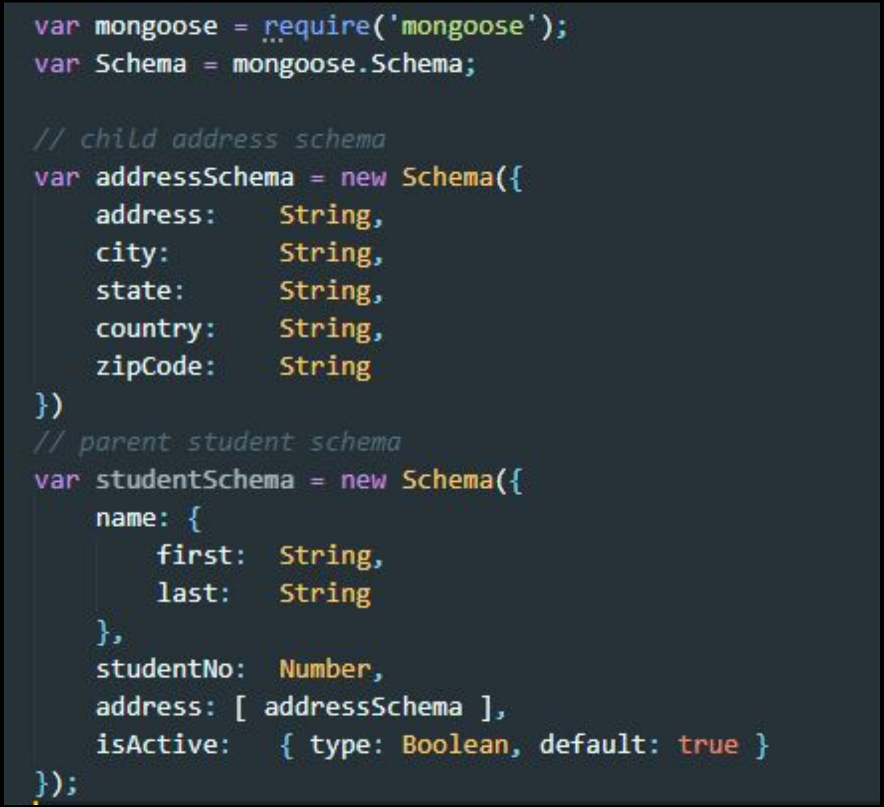
```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// simple schema, with a flat schema definition
var studentSchema = new Schema({
    name:        String,
    studentNo:   Number,
    address:     String,
    city:        String,
    state:       String,
    country:     String,
    zipCode:     String,
    isActive:    Boolean
});
```



```
// basic example.pass an object into Schema constructor
var simpleSchema = new Schema({ fieldName: SchemaType });
```

- While MongoDB is schema-less, SQL defines a schema via the table definition.
- A Mongoose 'schema' is a document data structure (or shape of the document) that is enforced via the application layer.
- Allowed Mongoose Data Types
    - String
    - Number
    - Date
    - Buffer
    - Boolean
    - Mixed
    - ObjectId
    - Array

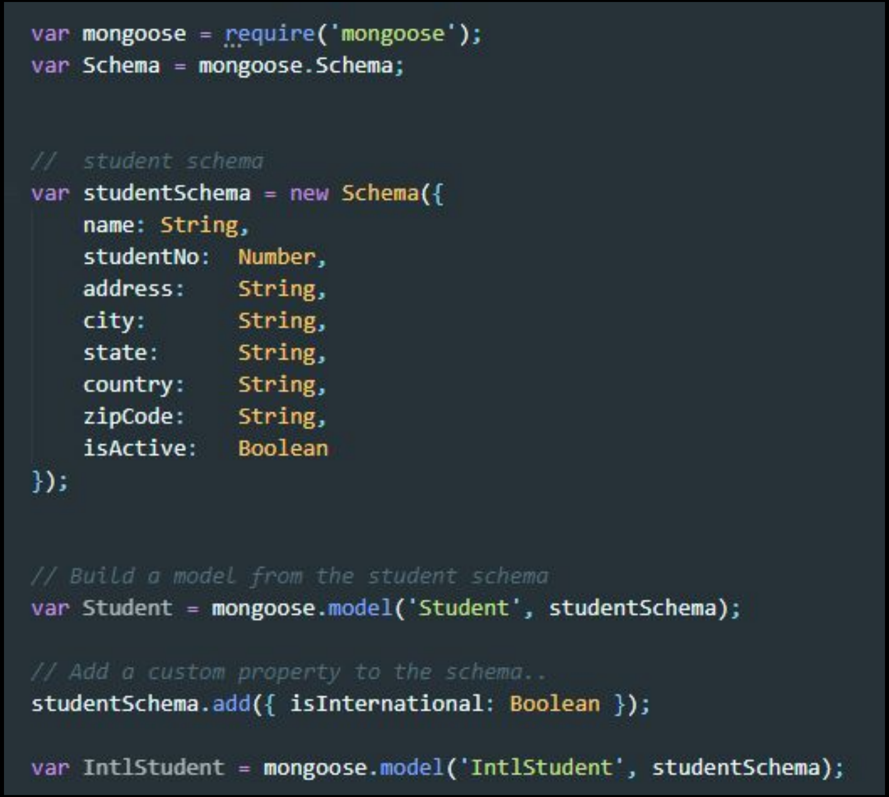
More Complex Schema



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

// child address schema
var addressSchema = new Schema({
    address:    String,
    city:       String,
    state:      String,
    country:    String,
    zipCode:    String
})
// parent student schema
var studentSchema = new Schema({
    name: {
        first:  String,
        last:   String
    },
    studentNo:  Number,
    address: [ addressSchema ],
    isActive:   { type: Boolean, default: true }
});
```

- Separate out nested objects into their own schemas ie. address schema for clarity
- Objects can be used as the data types and the default values can be set ie. isActive defaults = true
- We can have many levels of schema nesting,
  ie. addressSchema could have another schema countrySchema

Mongoose Model

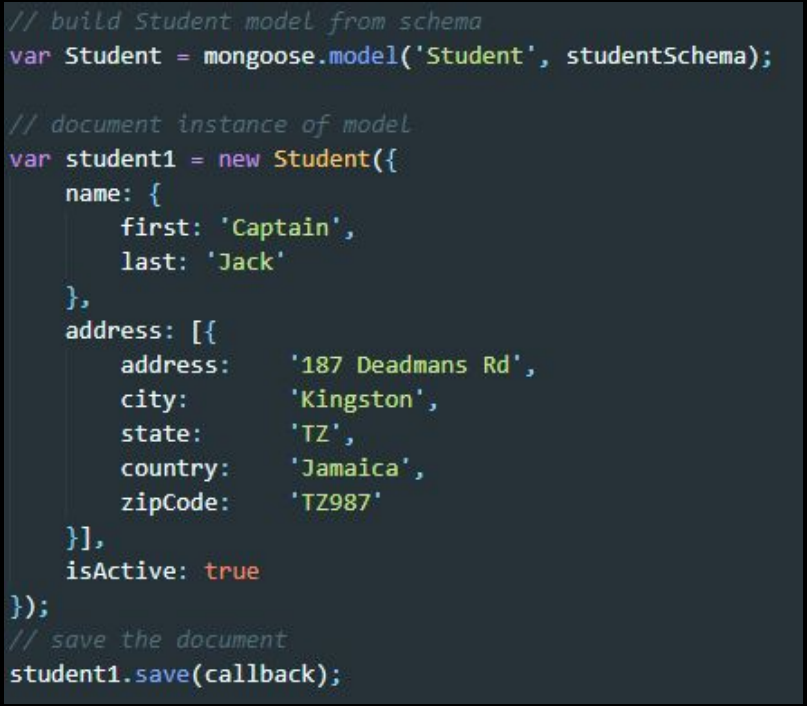


```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;


//  student schema
var studentSchema = new Schema({
    name: String,
    studentNo:  Number,
    address:    String,
    city:       String,
    state:      String,
    country:    String,
    zipCode:    String,
    isActive:   Boolean
});


// Build a model from the student schema
var Student = mongoose.model('Student', studentSchema);

// Add a custom property to the schema..
studentSchema.add({ isInternational: Boolean });

var IntlStudent = mongoose.model('IntlStudent', studentSchema);
```

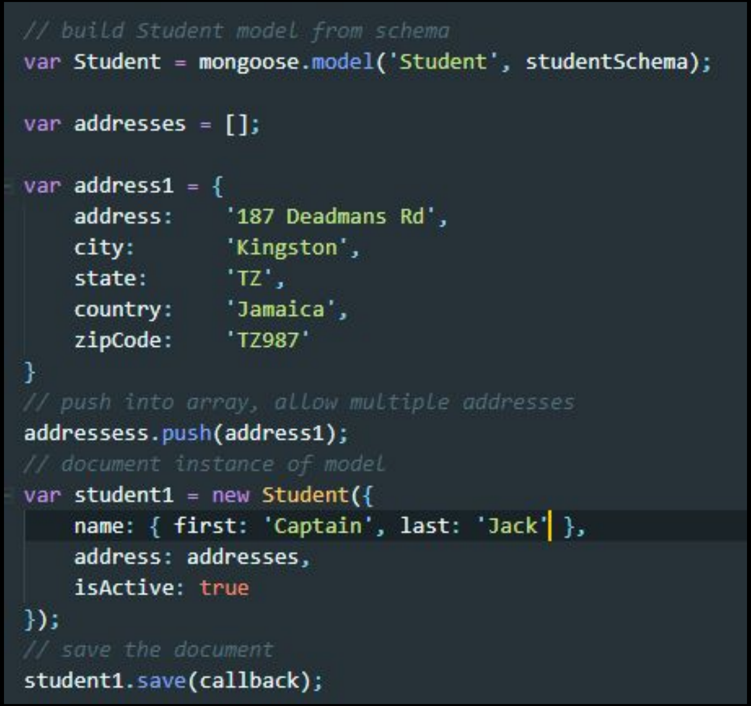
- A model is a wrapper on the schema, provide CRUD interface
- Models are higher-order constructor that take a schema and create an instance of a document equivalent to records in a relational database
- To create a Model, pass in the model name and the document schema to ctor
- More models can be built based on the schema definitions
- Properties can be appended to the schema to create a new schema

Mongoose Document & Sub Documents

A document is just an instance of our model. We can have multiple documents per model. Also, multiple sub documents per document.



```javascript
// build Student model from schema
var Student = mongoose.model('Student', studentSchema);

// document instance of model
var student1 = new Student({
    name: {
        first: 'Captain',
        last: 'Jack'
    },
    address: [{
        address:    '187 Deadmans Rd',
        city:       'Kingston',
        state:      'TZ',
        country:    'Jamaica',
        zipCode:    'TZ987'
    }],
    isActive: true
});
// save the document
student1.save(callback);
```



```javascript
// build Student model from schema
var Student = mongoose.model('Student', studentSchema);

var addresses = [];

var address1 = {
    address:    '187 Deadmans Rd',
    city:       'Kingston',
    state:      'TZ',
    country:    'Jamaica',
    zipCode:    'TZ987'
}
// push into array, allow multiple addresses
addressess.push(address1);
// document instance of model
var student1 = new Student({
    name: { first: 'Captain', last: 'Jack' },
    address: addresses,
    isActive: true
});
// save the document
student1.save(callback);
```