**Quote Keeper: Model Layer**

In this project, you will be completing the test suite for the Quote Keeper application. The suite currently has feature and server tests to verify that a user can enter quote information, post it to the server, and view the information in the returned webpage. Those tests are passing, but the app doesn’t organize and store the quote data anywhere.

To test-drive that functionality, we’ve added one failing server test. It will push you to the model layer, where you will begin the red, green, refactor cycle, setting up a database and defining a model using [MongoDB](https://www.mongodb.com/what-is-mongodb), Node’s [Mongoose](http://mongoosejs.com/) package, and the [Chai assertion module](http://chaijs.com/api/assert/).

Your suite will test code at every level of your application. When you write code in a new file, remember to import it to every layer so that your tests have access to it.

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**15/15Complete**

Mark the tasks as complete by checking them off

**Red at Server and Model**

**1.**

We have added a second server test that verifies that a “Quote” document is saved to the database.

Run it with npm test. Note the passing test and the failing error message:

ReferenceError: Quote is not defined

We are pushed to the model layer to define the Quote model.

**2.**

Welcome to the model layer, where you will write a test to drive the development of the Quote model!

In **test/models/quote-test.js**, import the Chai assertion module and Mongoose-related variables for working with a database by pasting the following code:

const {assert} = require('chai');

const {mongoose, databaseUrl, options} = require('../../database');

Remember to save your work after every step.

**3.**

Create a describe block for the Quote model.

Within the block, define a beforeEach hook that connects to the database and removes existing data by dropping the Database.

Define an afterEach hook that disconnects from the database.

Hint

describe('Quote', () => {

beforeEach(async () => {

await mongoose.connect(databaseUrl, options);

await mongoose.connection.db.dropDatabase();

});

afterEach(async () => {

await mongoose.disconnect();

});

});

**4.**

Below the hooks, create an inner describe - it block to verify that #quote is a String.

Within the block, create a new Quote with a quote attribute and assert that the returned instance has a quote attribute of the same String value.

You can create a new Quote with this:

const citation = new Quote({quote: ...});

Hint

describe('#quote', () => {

it('is a String', () => {

const quoteAsInt = 1;

const citation = new Quote({quote: quoteAsInt});

assert.strictEqual(citation.quote, quoteAsInt.toString());

});

});

**5.**

Run the new test. The test will fail because we have not implemented the code that connects our project to a database.

Uncomment the code in **database.js**, run npm install mongoose in the terminal, then run the test suite again.

**6.**

The test will fail for a different reason: The model is still not defined!

In the **models/quote.js** file, import mongoose from **database.js** and create a Mongoose Schema with one path named quote of type String.

Hint

const {mongoose} = require('../database');

const quoteSchema = new mongoose.Schema({

quote: String

});

**7.**

Compile your Schema to a model using mongoose.model() and export the model by using the expression module.exports = YourModelName in **quotes.js**.

Import your model in **quote-test.js**.

Hint

In **models/quotes.js**

const Quote = mongoose.model('Quote', quoteSchema);

module.exports = Quote;

At the top of **test/models/quote-test.js**

const Quote = require('../../models/quote');

**Green and Refactor at Model**

**8.**

Run the test, see that the model test passes, then refactor by extracting hook methods to helper functions in **database.js**. (We’ll handle the server errors soon.)

To begin refactoring, create two functions in **database.js** that execute the same method calls as the beforeEach and afterEach hooks in **quote-test.js**.

Export them at the end of the file.

Hint

Delete the old module.exports expression in **database.js**, then paste this at the end.

const connectAndDrop = async () => {

await mongoose.connect(databaseUrl, options);

await mongoose.connection.db.dropDatabase();

};

const disconnect = async () => {

await mongoose.disconnect();

};

module.exports = {

mongoose,

databaseUrl,

options,

connectAndDrop,

disconnect,

};

**9.**

In **test/models/quote-test.js**, import the new helper functions and use them to replace the current expressions in your hooks.

Run the test. Make sure the model test passes before moving back to the server layer.

Hint

At the top of the file, replace the old **database.js** import with

const {connectAndDrop, disconnect} = require('../../database');

In the describe block replace the old hooks with

beforeEach(connectAndDrop);

afterEach(disconnect);

**Red at Server**

**10.**

Now that you have incorporated mongoDB and Mongoose into your project, you can use the new Quote model and the database in your server tests.

Import the model file, database file, and connect/disconnect (via hooks) in your **test/routes/index-test.js** file.

Hint

Towards the top of your file add:

const Quote = require('../../models/quote');

const {connectAndDrop, disconnect} = require('../../database');

Within the first describe add:

beforeEach(connectAndDrop);

afterEach(disconnect);

**11.**

Run the tests to review the server level error message: a TypeError about the quote path. No Quote instance is being created at the server level.

Import the Quote model and uncomment the code in **routes/index.js**. This creates a Quote instance every time the server receives a POST request to the root, called /.

Hint

Add this towards the top:

const Quote = require('../models/quote');

Make sure this line is uncommented:

await Quote.create({quote, attributed, source});

**12.**

Run the test suite — but you still see red! We’re getting a new error about the #attributed property (Mary Shelley in this case) because we haven’t defined it in the model.

**Red at Model**

**13.**

In your **models/quote-test.js** file below the #quote test, add similar tests for the #attributed and #source properties of the Quote model and run them.

* These two tests should force you to define additional paths in the model. For each one, create a new Quote instance with the desired path, then verify that its new path is type String.
* Chai’s [isString](http://chaijs.com/api/assert/#method_isstring) and [strictEqual methods](http://chaijs.com/api/assert/#method_strictequal) will be useful here.

Hint

describe('#attributed', () => {

it('is a String', () => {

const attributed = 'James Baldwin';

const citation = new Quote({attributed});

assert.isString(citation.attributed);

assert.strictEqual(citation.attributed, attributed);

});

});

describe('#source', () => {

it('is a String', () => {

const source = 'Notes of a Native Son';

const citation = new Quote({source});

assert.isString(citation.source);

assert.strictEqual(citation.source, source);

});

});

**14.**

Get your tests passing by adding two paths to the Schema in **models/quotes.js**.

Hint

Add this below the quote path:

attributed: String,

source: String

**Green at Server and Model**

**15.**

Run the test suite and see your work pay off — your tests are passing at the model and server layers!

But the feature test is now failing. As in earlier steps, you need to give your feature test access to the database and Mongoose.

Uncomment the three sections of code in **wdio.conf.js**. This is the configuration file for feature tests.

Although it’s not used here, npm start is affected as well. Uncomment the two sections of code in **bin/www**, the file used when you run that command.

Hint

Uncomment these lines in **wdio.conf.js**:

const {connectAndDrop, disconnect} = require('./database');

connectAndDrop();

disconnect();

Uncomment these lines in **bin/www**:

const {mongoose, databaseUrl, options} = require('../database');

mongoose.connect(databaseUrl, options).then(()=> {

server.listen(port);

});