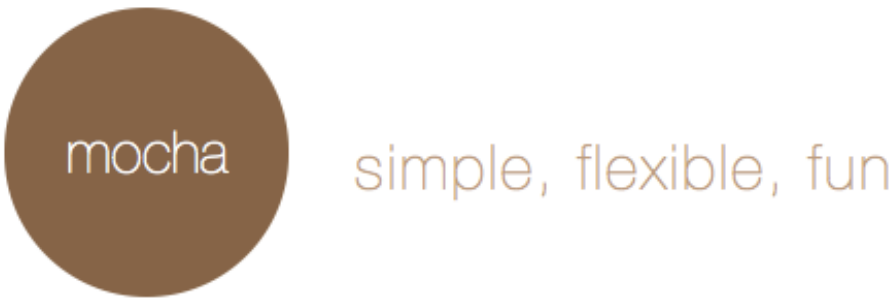
webapplog: [programming weblog]

Software engineering, Node.js, JavaScript and startups.

Test-Driven Development in Node.js With Mocha

Who needs Test-Driven Development?

Imagine that you need to implement a complex feature on top of an existing interface, e.g., a 'like' button on a comment. Without tests you'll have to manually create a user, log in, create a post, create a different user, log in with a different user and like the post. Tiresome? What if you'll need to do it 10 or 20 times to find and fix some nasty bug? What if your feature breaks existing functionality, but you notice it 6 months after the release because there was no test!



Mocha: simple, flexible, fun

Don't waste time writing tests for throwaway scripts, but please adapt

the habit of Test-Driven Development for the main code base. With a little time spent in the beginning, you and your team will save time later and have confidence when rolling out new releases. Test Driven Development is a really really really good thing.

Quick Start Guide

Follow this quick guide to set up your Test-Driven Development process in Node.js with Mocha.

Install Mocha globally by executing this command:

```
$ sudo npm install -g mocha
```

We'll also use two libraries, Superagent and expect.js by LeanBoost. To install them fire up npm commands in your project folder like this:

```
$ npm install superagent
$ npm install expect.js
```

Open a new file with `.js` extension and type:

```
var request = require('superagent');
var expect = require('expect.js');
```

So far we've included two libraries. The structure of the test suite going to look like this:

```
describe('Suite one', function(){
  it(function(done){
  ...
  });
  it(function(done){
  ...
  });
});
describe('Suite two', function(){
  it(function(done){
  ...
  });
});
```

Inside of this closure we can write request to our server which should be running at localhost:8080:

```
...
it (function(done){
  request.post('localhost:8080').end(function(res){
    //TODO check that response is okay
  });
});
...
```

Expect will give us handy functions to check any condition we can think

of:

```
...
expect(res).to.exist;
expect(res.status).to.equal(200);
expect(res.body).to.contain('world');
...
```

Lastly, we need to add **done()** call to notify Mocha that asynchronous test has finished its work. And the full code of our first test looks like this:

```
var request = require('superagent');
var expect = require('expect.js');

describe('Suite one', function(){
 it (function(done){
   request.post('localhost:8080').end(function(res){
    expect(res).to.exist;
    expect(res.status).to.equal(200);
    expect(res.body).to.contain('world');
    done();
   });
  });
});
```

If we want to get fancy, we can add **before** and **beforeEach** hooks which will, according to their names, execute once before the test (or suite) or each time before the test (or suite):

```
before(function(){
  //TODO seed the database
});
describe('suite one ',function(){
  beforeEach(function(){
    //todo log in test user
  });
  it('test one', function(done){
  ...
  });
});
```

Note that before and beforeEach can be placed inside or outside of describe construction.

To run our test simply execute:

```
$ mocha test.js
```

To use different report type:

```
$ mocha test.js -R list
$ mocha test.js -R spec
```

This entry was posted in Advices, Education, JavaScript, Node.js, Tutorials and tagged expect.js, JavaScript, learnboost, mocha, mocha.js, Node, node.js, quick start guide, superagent, tdd, test-driven development on March 15, 2013 [http://webapplog.com/test-driven-development-in-node-js-with-mocha/] .

11 thoughts on “Test-Driven Development in Node.js With Mocha”

Aidan

September 28, 2014 at 10:01 am

Thanks a lot, Azat! It was a very good intro to TDD in Node.js!

Azat Post author
September 2, 2014 at 12:30 pm

I'm glad it was helpful. Yes, often the documentation is lacking.

Nomikos
September 1, 2014 at 12:28 pm

Thank you! In one paragraph you have managed to explain what Mocha actually *IS FOR*, something I could not grasph from their homepage. This happens way too often on newer front-end tools.. you need to know what it is and how it works before you can even understand the manual/documentation..

Pingback: MEAN 2

Pingback: mocha를 이용한 node.js | Beagle Bone Black with node.js

Azat Post author
May 31, 2013 at 6:58 pm

Great, thanks! That's what happens when I write code in the ByWord app. :-)

Marcos
May 31, 2013 at 3:09 pm

There's also a typo in the last mocha command :D

Pingback: Final Research Links | Recipe Finder

Azat Post author
March 25, 2013 at 1:53 pm

@Jay,
Thanks! That's what happens when you code in a text editor. :-)

Jay Scott ANDERSON
March 24, 2013 at 12:02 pm

The example is apparently missing closing brackets, "request.post('localhost:8080').end(" not closed.

Pingback: Faysal Ahmed