Testing

Software Engineering - Lab

Marco Robol - marco.robol@unitn.it

Testing with Jest

> A JavaScript Testing Framework with a focus on simplicity - jestjs.io

Install Jest `npm install --save-dev jest`

- Create `sum.js` that exports a `sum()` function; then in `sum.test.js` :

```
test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

> https://jestjs.io/docs/en/getting-started.html

- To run `jest` , create an npm script `test` in `package.json:`

```
"test": "NODE_OPTIONS=--experimental-vm-modules jest",
```

> To use Jest with ESModules, use `NODE_OPTIONS=--experimental-vm-modules`
>
> https://jestjs.io/docs/ecmascript-modules

- Run with `npm run test`

Matchers jestjs.io/docs/using-matchers

`./someModule.js`

```
function concatenateStrings (a, b) { return '' + a + b }
module.exports = concatenateStrings
```

`./someModule.test.js`

```
const conc = require('./someModule')
test('conc(2,2)', () => {
    expect(conc(2, 2)).toBe('22');
});
test('concat("a","b")', () => {
    expect(conc('a','b')).toBe('ab');
});
test('concat(null,null)', () => {
    expect(conc(null,null)).toBe('nullnull');
});
```

Asynchronous Code Testing jestjs.io/docs/asynchronous

> When you have code that runs **asynchronously**, Jest needs to know when the code it is testing has completed, before it can move on - https://jestjs.io/docs/asynchronous

`npm install --save-dev node-fetch` https://www.npmjs.com/package/node-fetch

```
const fetch = require("node-fetch");
const url = process.env.API_URL || "https://easy-lib.onrender.com/api/v1"
it('works with get', async () => {
    expect.assertions(1)
    expect( ( await fetch(url+"/books") ).status ).toEqual(200)
})
```

Testing an API with `node-fetch`

`npm install --save-dev node-fetch` https://www.npmjs.com/package/node-fetch

```
// ./api.test.js
const fetch = require("node-fetch");
const url = process.env.API_URL || "https://easy-lib.onrender.com/api/v1"
it('works with get', async () => {
    expect.assertions(1)
    expect( ( await fetch(url+"/books") ).status ).toEqual(200)
})
it('works with post', async () => {
    expect.assertions(1)
    var response = await fetch(url+'/books', {
        method: 'POST', body: JSON.stringify({title: 'Testing with jest'}),
        headers: { 'Content-Type': 'application/json' }
    })
    expect( ( await response.json() ).status ).toEqual(201)
})
```

Testing an API with `supertest`

`npm install --save-dev supertest` https://www.npmjs.com/package/supertest

```
// EasyLib\app\app.test.js
const request = require('supertest');
const app     = require('./app');

test('app module should be defined', () => {
  expect(app).toBeDefined();
});

test('GET / should return 200', () => {
  return request(app)
    .get('/')
    .expect(200);
});
```

TODO: Create a simple Express.js application in app.js that replies with `200` at `GET /`

Configuring Jest

Simple preloading *dotenv*: in `test` *script* within `package.json` and run with `npm test`

```
                                                                    // package.json
"dev": "node -r dotenv/config index.js",      // dotenv preloaded
"test": "jest --setupFiles dotenv/config"     // dotenv preloaded
```

Or, configuration file `jest.config.js` https://jestjs.io/docs/en/configuration.html

```
                                                                    // jest.config.js
module.exports = {
  setupFiles: ["<rootDir>/.jest/setEnvVars.js"],
  verbose: true
```

```
                                                                    // .jest/setEnvVars.js
require("dotenv").config()
```

Testing *token-authenticated* APIs & *db connection* with supertest

```
// EasyLib\app\booklendings.test.js
const request  = require('supertest');    const app      = require('./app');
const jwt      = require('jsonwebtoken'); const mongoose = require('mongoose');
describe('POST /api/v1/booklendings', () => {

  beforeAll( async () => {                              // establish connection to db
    jest.setTimeout(8000);
    app.locals.db = await  mongoose.connect(process.env.DB_URL); });
  afterAll( () => { mongoose.connection.close(true); });

  var token = jwt.sign( {email: 'John@mail.com'},
    process.env.SUPER_SECRET, {expiresIn: 86400} ); // create a valid token

  test('POST /api/v1/booklendings with Student not specified', () => {
    return request(app).post('/api/v1/booklendings')
    .set('x-access-token', token).set('Accept', 'application/json')
    .expect(400, { error: 'Student not specified' });
  });
```

TODO: Create a mongodb-based `app.js` and *setup connection* when `supertest` !

Test EasyLib with *mock-functions*

`EasyLib\app\books.test.js` https://jestjs.io/docs/en/mock-functions

```
describe('GET /api/v1/books', () => {
  let bookSpy; // Moking Book.find method
  beforeAll( () => {
    const Book = require('./models/book');
    bookSpy = jest.spyOn(Book, 'find').mockImplementation((criterias) => {
      return [{ id: 1010, title: 'Jest' }];
    });
  });
  afterAll(async () => { bookSpy.mockRestore(); bookSpyFindById.mockRestore(); });

  test('GET /api/v1/books should respond with an array of books', async () => {
    request(app).get('/api/v1/books').expect('Content-Type', /json/).then( (res) => {
        if(res.body && res.body[0])
          expect(res.body[0]).toEqual({self:'/api/v1/books/1010',title:'Jest'})
    });
  });
});
```

TODO: Try the to *mock* a function

EasyLib Links

EasyLib repos

BackEnd - https://github.com/unitn-software-engineering/EasyLib

Vue FrontEnd - https://github.com/unitn-software-engineering/EasyLibVue

EasyLib deploys

Basic Frontend - https://easy-lib.onrender.com/

Vue Frontend - https://easy-lib.onrender.com/EasyLibApp/ or https://unitn-software-engineering.github.io/EasyLibApp/

Questions?

marco.robol@unitn.it

Coverage

Configure Jest to activate **coverage** - jestjs.io/docs/configuration:

```
// package.json
"jest": {
    "verbose": true,
    "collectCoverage": true
}
```

or

```
// jest.config.json
{
  "verbose": true
}
```