Testing - Deployment - Continuous Integration

Software Engineering, 2nd part - Lab

Marco Robol - marco.robol@unitn.it

Academic year 2021/2022 - Second semester

Plan for the second part of the course

- April 19-21: Design thinking, project arch, API
- April 26-28: Foundations JS, Node.js, git
- May 2-5: Agile Methodology, MongoDB, API
- May 9 - May 22: Sprint #1
  - More on agile methodology, testing, git branching
- ***May 23 - June 7: Sprint #2***
  - More on testing, devops/CI

Contents of today class

In today's class, we will see how to use Jest to test our APIs, how to set-up a continuous integration environment with Travis CI, and how to deploy our application on Heroku.

> **EasyLib repositories**
> WebAPIs - https://github.com/unitn-software-engineering/EasyLib - `npm run dev`
> FrontEnd - https://github.com/unitn-software-engineering/EasyLibVue

> **EasyLib deployment**
> Basic Frontend - https://easy-lib.herokuapp.com/
> Vue Frontend - https://unitn-software-engineering.github.io/EasyLibApp/

Testing with Jest

Jest is a delightful JavaScript Testing Framework with a focus on simplicity - jestjs.io

Install Jest in develompment environment and run it

1. `npm install --save-dev jest`
2. Create a `sum.test.js`

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

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

3. Run `jest`

Testing a function *concatenateStrings(a, b)*

`./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 test', () => {
    expect(conc('a','b')).toBe('ab');
});
test('concat null', () => {
    expect(conc(null,null)).toBe('nullnull');
});
```

Testing an API with `node-fetch`

`./api.test.js`

```
const fetch = require("node-fetch");
const url = process.env.HEROKU || "http://localhost:3000"
it('works with get', async () => {
    expect.assertions(1)
    expect( ( await fetch(url) ).status ).toEqual(200)
})
it('works with post', async () => {
    expect.assertions(1)
    var response = await fetch(url+'/courses', {
        method: 'POST', body: JSON.stringify({name: 'hello course'}),
        headers: { 'Content-Type': 'application/json' }
    })
    expect( ( await response.json() ).status ).toEqual(201)
})
```

`npm install --save-dev node-fetch`: This requires the server to be running!

Testing an API with `supertest`

> https://www.npmjs.com/package/supertest `npm install --save-dev 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);
});
```

Configuring Jest on EasyLib using *dotenv module*

> https://lusbuab.medium.com/using-dotenv-with-jest-7e735b34e55f

1. Add *test* **script** to `package.json`:

```
"scripts": {
    "start": "node api.js",                          // dotenv not preloaded
    "dev": "node -r dotenv/config index.js",         // dotenv preloaded
    "test": "jest --setupFiles dotenv/config"        // dotenv preloaded
```

2. Run jest `npm test`

Alternatively...

Configure jest to load *environment variables* from `.env` without preloading *dotenv module*

1. In `package.json` do not preload dotenv `"test": "jest"`
2. From https://jestjs.io/docs/en/configuration.html, create `jest.config.js` and set:

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

2. Create file `./.jest/setEnvVars.js` to load dotenv:

```
require("dotenv").config()
```

Testing EasyLib *token-authenticated* APIs on *mongodb*

`EasyLib\app\booklendings.test.js`

```
const request  = require('supertest');    const app      = require('./app');
const jwt      = require('jsonwebtoken'); const mongoose = require('mongoose');

describe('GET /api/v1/booklendings', () => {

  beforeAll( async () => { 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' });
  });
```

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'})
    });
  });
});
```

Coverage

Configure Jest to activate **coverage**

> https://jestjs.io/docs/configuration#collectcoverage-boolean:

`package.json`

```
"jest": {
    "verbose": true,
    "collectCoverage": true
}
```

Run tests with `npm run test`

Heroku

Cloud Application Platform - Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud - heroku.com

Setup Heroku CLI on your your machine

https://devcenter.heroku.com/articles/getting-started-with-nodejs?singlepage=true

1. Register on https://www.heroku.com/
2. Install Heroku CLI https://devcenter.heroku.com/articles/heroku-cli
3. CLI log in `heroku login`

Prepare the application

https://devcenter.heroku.com/articles/preparing-a-codebase-for-heroku-deployment#

1. Create *Procfile* `.\Procfile`:

   https://devcenter.heroku.com/articles/preparing-a-codebase-for-heroku-deployment#3-add-a-procfile

```
web: node index.js
test: jest
```

2. Set listening port of Node server:

   https://devcenter.heroku.com/articles/preparing-a-codebase-for-heroku-deployment#4-listen-on-the-correct-port

```
const PORT = process.env.PORT || 8080
```

Create heroku app and connect with local repository.

https://devcenter.heroku.com/articles/git#creating-a-heroku-remote

1. **Create a new Heroku app** from your existing repository `heroku create`

   Alternatively, create a new app from the Heroku Dashboard and manually add remote source to your local repository `heroku git:remote -a our-heroku-app`

2. **Configure env vars** from *Setting->Config Vars* e.g. DB_URL and SUPER_SECRET

3. **Push** repository on *heroku* remote source `git push heroku main`

4. **Start the Heroku app** `heroku ps:scale web=1` and open it `heroku open` or view logs `heroku logs --tail`

You can also run the Heroku app locally through the Procfile `heroku local web`

Travis CI

> The simplest way to test and deploy your projects in the cloud. Easily sync your projects with Travis CI and you'll be testing your code in minutes - travis-ci.org

Configure repository to use Travis CI

> https://docs.travis-ci.com/user/tutorial/#to-get-started-with-travis-ci-using-github

- Create file `.travis.yml` :

```
language: node_js
```

- On Travis CI Dashboard add your repository
- Check build status of your application
- Click on "Build status image" build passing to get the markdown code
- Embed "Build status image" build passing in your `readme.MD`

> https://travis-ci.org/github/unitn-software-engineering/EasyLib build passing

`.gitignore` - Ignoring files from git versioning

- You can start from generic `.gitignore` file generated on www.gitignore.io, such as, https://www.gitignore.io/api/node,windows,linux,visualstudiocode
- **Make sure to always ingore**: `node_modules` `coverage` `.env`
- Put the `.gitignore` file itself under version control `git add .gitignore`

Automatically deploy your Heroku application

- Deploy automatically on Heroku after a successful build by Travis CI:

  https://docs.travis-ci.com/user/deployment/heroku/

  `.travis.yml`:

```yaml
deploy:
  provider: heroku
  api_key:
    secure: "YOUR ENCRYPTED API KEY"
```

- Alternatively, auto-deployment from GitHub can be configured on Heroku Dashboard:

  https://devcenter.heroku.com/articles/github-integration

Front-end deployment

> EasyLib Front-end Repository
> https://github.com/unitn-software-engineering/EasyLibVue

> EasyLib deployed front-end application
> https://unitn-software-engineering.github.io/EasyLibApp/

Frontend already in the same repository as your webAPIs

`EasyLib\app\app.js`

```
// Serving frontend files from process.env.FRONTEND
app.use('/', express.static(process.env.FRONTEND || 'static'));
// If request not handled, try in ./static
app.use('/', express.static('static'));
// If request not handled, try with next middlewares ...
```

`EasyLib\.env` These configurations are used only locally, never commit these settings!

```
# Path to external frontend - If not provided, basic frontend in static/index.html is used
FRONTEND='../EasyLibVue/dist'
```

Separately setup Heroku with appropriate environment variables!

Build and serve Vue app on *GitHubPages* pages.github.com

When ready to ship app to production, run the following: `npm run build`. This generates minified html+javascript frontend in `.\dist` folder. Create a **dedicated repository for hosting** on github to host your frontend, then push your built frontend manually or with a script `EasyLibVue\deploy.sh`:

```
npm run build # build Vue app
cd dist # navigate into the build output directory
git init
git add -A
git commit -m 'deploy'
git push -f https://github.com/unitn-software-engineering/EasyLibApp.git master:gh-pages
```

MUST be a repository dedicated to hosting, different from your Frontend repository!

Run `.\deploy.sh` (In case of errors, manually delete the folder `.\dist`).

> https://cli.vuejs.org/guide/deployment.html#github-pages

Questions?

marco.robol@unitn.it