Testing

Tests unitarios en Node.js

Álvaro Alonso

Tests unitarios en aplicaciones Web

- **Funcionalidades típicas a probar**
  - Rutas de la aplicación
  - Lógica de negocio
  - Renderización de vistas
  - Operaciones que leen/modifican la base de datos

- Tecnologías
  - **Mocha**
    - Framework de pruebas de JavaScript que se ejecuta en Node.js.
  - **Librerías de aserciones**
    - assert – Módulo incluido en Node.js
    - should.js – expect.js – librerías basadas en aserciones should y expect
    - **chai** – soporta aserciones tipo assert, should y expect

Mocha

- Interfaz para la definición de tests
  - **describe**(): define una batería de tests unitarios
    - Pueden concatenarse
  - **it**(): define cada test unitario
    - Un test no pasa si el código lanza un error
    - En caso contrario pasa y se continua con el siguiente
- Hooks
  - **before**(): código a ejecutar antes de una batería de tests
  - **after**(): código a ejecutar antes de una batería de tests
  - **beforeEach**(): código a ejecutar antes de cada test unitario
  - **afterEach**(): código a ejecutar después de cada test unitario

Mocha – ciclo de ejecución hooks

mocha_bdd.js

```
1  before(function () {  console.log(' - Before all tests'); });
2  after(function () {  console.log(' - After all tests');   });
3
4  describe('Test group 1', function () {
5
6    before(function () {  console.log(' - Before test group 1');  });
7    after(function () {  console.log(' - After test group 1');  });
8
9    beforeEach(function () {  console.log(' - Before each unit test of group 1'); });
10   afterEach(function () {  console.log(' - After each unit test of group 1'); });
11
12   it('Unit test 1 should not fail', function () {
13     console.log(' - Unit test 1');
14   });
15
16   it('Unit test 2 should not fail', function () {
17     console.log(' - Unit test 2');
18   });
19 });
20
21 describe('Test group 2', function () {
22
23   before(function () {  console.log(' - Before test group 2');  });
24   after(function () {  console.log(' - After test group 2');  });
25
26   it('Unit test 3 should fail', function () {
27     console.log(' - Unit test 3');
28     throw new Error('Unit 3 test error');
29   });
30 });
```

Mocha – ciclo de ejecución hooks

test.js

```
1  before(function () {  console.log(' - Before all tests'); });
2  after(function () {  console.log(' - After all tests');   });
3
4  describe('Test group 1', function () {
5
6    before(function () {  console.log(' - Before test group 1
7    after(function () {  console.log(' - After test group 1')
8
9    beforeEach(function () {  console.log(' - Before each uni
10   afterEach(function () {  console.log(' - After each unit
11
12   it('Unit test 1 should not fail', function () {
13     console.log(' - Unit test 1');
14   });
15
16   it('Unit test 2 should not fail', function () {
17     console.log(' - Unit test 2');
18   });
19 });
20
21 describe('Test group 2', function () {
22
23   before(function () {  console.log(' - Before test group 2
24   after(function () {  console.log(' - After test group 2')
25
26   it('Unit test 3 should fail', function () {
27     console.log(' - Unit test 3');
28     throw new Error('Unit 3 test error');
29   });
30 });
```

```
- Before all tests
 Test group 1
- Before test group 1
- Before each unit test of group 1
- Unit test 1
   ✓ Unit test 1 should not fail
- After each unit test of group 1
- Before each unit test of group 1
- Unit test 2
   ✓ Unit test 2 should not fail
- After each unit test of group 1
- After test group 1

 Test group 2
- Before test group 2
- Unit test 3
   1) Unit test 3 should fail
- After test group 2

- After all tests

 2 passing (12ms)
 1 failing

 1) Test group 2
      Unit test 3 should fail:
    Error: Unit test 3 error
     at Context.<anonymous> (mocha_bdd.js:28:11)
```

Mocha – set up

Instalación paquete npm y ejecución fichero de pruebas

- Instalación global

```
$ npm install -g mocha
$ mocha test.js
```

- Instalación directorio de desarrollo

```
$ npm install --save-dev mocha
$ ./node_modules/mocha/bin/mocha test.js
```

- Test script en package.json
  - Ejecuta tests en test/*.js o en test.js

package.json

```
{
  "scripts": {
    "test": "mocha"
  }
}
```

```
$ npm test
```

Mocha – tests inclusivos y exclusivos

- Se aplican sobre *describe* o *it*
  - skip() – no se ejecuta este test o conjunto de tests
    - Puede aplicarse en tiempo de ejecución usando this.skip()
  - only() – solo se ejecuta este test o conjunto de tests

```
describe.only('Test group 1', function () {

  it('Unit test 1 should not fail', function () {
    // ...
  });

  it('Unit test 2 should not fai
    // ...
  });
});

describe('Test group 2', functio

  it('Unit test 3 should fail',
    // ...
  });
});
```

```
describe('Test group 1', function () {

  it('Unit test 1 should not fail', function () {
    if ( /* condition */ ) {
      // assertions
    } else {
      this.skip();
    }
  });

  it.skip('Unit test 2 should not fail', function () {
    // ...
  });
});
```

Mocha – tests asíncronos

- Por defecto, los tests son síncronos
  - Una vez acabado un test, Mocha pasa al siguiente
- Tres alternativas para finalizar tests asíncronos (aplica también a hooks)
  - Llamar al callback done()

```
1  it('should pass without error', function (done) {
2    myAsyncFunction(function (err) {
3      if (err) done(err);       // El test no pasa
4      else done();              // El test pasa
5    });
6  });
```

  - Devolver una promesa

```
1  it('should pass without error', function () {
2    return new Promise(function (resolve) {
3      resolve();          // El test pasa si se resuelve la promesa
4    });
5  });
```

  - Usar async / await

```
1  it('should pass without error', async function () {
2    let res = await myAsyncFunction();
3    if (!res) throw new Error('Test fail');   // El test falla si se lanza un error. Si no
4  });                                         // pasa y se continúa con el siguiente
```

Timeouts

- En tests asíncronos puede especificarse un timeout
  - A nivel de conjunto de tests
  - A nivel de test unitario
  - A nivel de hook

```
1   describe('Test group 1', function () {
2     this.timeout(700);
3
4     before(function () {
5       this.timeout(500);
6       // ...
7     });
8
9     it('Unit test 1 should not fail', function () {
10      this.timeout(300);
11      // ...
12    });
13  });
```

Chai

- Soporta aserciones tipo **assert**, **should** y **expect**
- Puede integrarse en cualquier framework de testing
  - Incluido Mocha
- Una aserción lanza un error si no se cumple
  - En Mocha haciendo que no se cumpla el test correspondiente
- Instalación

```
$ npm install chai
```

Tipos de aserciones en chai

chai_test.js

```
1  var foo = 'bar';
2
3  // Assert:
4  var assert    = require("chai").assert;
5
6  assert.typeOf( foo, 'string', 'foo is a string' );
7  assert.equal( foo, 'bar', 'foo equal `bar`' );
8  assert.lengthOf( foo, 3, 'foo`s value has a length of 3' );
9
10 // Expect:
11 var expect    = require("chai").expect;
12
13 expect( foo ).to.be.a( 'string' );
14 expect( foo ).to.equal( 'bar' );
15 expect( foo ).to.have.length( 3 );
16
17 // Should:
18 var should    = require("chai").should();
19
20 foo.should.be.a( 'string' );
21 foo.should.equal( 'bar' );
22 foo.should.have.length( 3 );
23
```

Documentación

- **Mocha**
  - https://mochajs.org
- **Chai**
  - https://www.chaijs.com

- Assert
  - https://nodejs.org/api/assert.html
- Should
  - https://github.com/shouldjs/should.js
- Expect
  - https://github.com/Automattic/expect.js

Testing

Tests unitarios en Node.js

Álvaro Alonso