Stella Technology

Exchange › Coordinate › Collaborate

Mocha

About Us

Amir Shahzad
QA Lead
Stella Technology

LinkedIn:
https://www.linkedin.com/in/amir-khan-687455b/

Twitter: @meetaamir

Agenda

1. What is a REST service?
2. Structure of Rest services
3. Testing of Rest services
4. Authoriztion /Authincation
5. What is Mocha Framework?
6. Why Mocha?
7. Super test agent
8. Chai Assertions.
9. General Tips

RESTful API

- A **RESTful API** is an application program interface (**API**) that uses HTTP requests to GET, PUT, POST and DELETE data. Representational state transfer (**REST**), which is used by browsers, can be thought of as the language of the Internet.

Structure of a REST service

1. End Point
   /user/login
2. Method Type
   GET/POST/PUT/DELETE…
3. Body (Request payload)
   content type = application/JSON
   {“email” : “test@test.com” , “password” : “test” }

Testing of a Rest service

- What is required ?
  - End point of the service
  - Method Type
  - Body
    - Request payload
    - Request headers

Assertions on the response

- Comparisons :
  - Expected output == actual output

Request body:

{“email” : “hi2344@com” , “password” : “test”} // Invalid email address – Login

Response body

{

error : Invalid email address

}

Mocha as Our Primary REST Testing Tool

- Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases
- Runs on Node.js/Browser
- Supports BDD/TDD
- Support for multiple assertion libraries (CHAI)
- Support for mocking library
- Async and promise support
- Highlights slow tests
- Support for custom test execution flow

Example

```
describe(‘Add', function () {

it(‘Add two Positive numbers', (done) => {

// test assertions
done()

      })
   })
it(‘Add two Negative numbers', (done) => {

// test assertions
done()

      })
   })
})
```

Mocha

- In mocha you can define test cases hierarchy using describe and it blocks.
  - both can be mixed but as general rule of thumb describe provides reporting hierarchy and 'it' defines the individual test cases.
  - describe and it both are internally executed in async flow.
  - all blocks like describe, it, before, after, AfterEach, beforeEach are executed in the order as suggested by their names.
  - Our actulal test is in the it block we normallu use an api wrapper that is a supertest agent instance
  - the final end() hook of super test is executed when an API is accessed and its response is received
  - - if the done method is invoked which is mocha's observer, if it is invoked then mocha will consider that test has been passed.
  - - if any assertion fails then that assertion will throw an exception and done method will not be invoked in that case mocha will consider this test as failing

Chai Assertion Library

- Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

```
AssertAssert
var assert = chai.assert;

assert.typeOf(foo, 'string');
assert.equal(foo, 'bar');
assert.lengthOf(foo, 3)
assert.property(tea, 'flavors');
assert.lengthOf(tea.flavors, 3);
```

Why have we decided to use Mocha?

1. Because the tests are written in a programming language, they are much more version control friendly
2. Because the tests are written in java script, we can avoid boilerplate by taking advantage of our existing code and OO principles
3. There is no UI to deal with
4. The tests can be run from command line, meaning we can also integrate them into Bamboo or other automated deployments