Testing using Cypress in JavaScript

Overview

In this tutorial, we will create some tests in JavaScript using Cypress.

Cypress is an automation framework focused on building and running browser enabled tests, either for unit, integration or E2E testing.

Of the many features provided by Cypress, the ability to run tests faster by running them directly in the browser, instead of sending commands to the browser through the network, is one of its strongest points.

Requirements

- nodejs
- "cypress", "mocha", "mocha-junit-reporter" node modules

Description

This example is mostly taken from the sample tutorial in Cypress documentation page.

You can use different reporters along with Cypress, including mochawesome; you may also use multiple reporters at the same time, in case you need to generate JSON, HTML reports, for example, simultaneously.

In this case we'll configure Cypress to use the "junit" reporter, which in turn seems to be using "mocha-junit-reporter" node module.

First, we need to configure our project properly, by defining the `package.json` content.

package.json

```
{
  "name": "cypress_tutorial",
  "version": "1.0.0",
  "description": "cypress tutorial",
  "main": "index.js",
  "scripts": {
    "test": "./node_modules/cypress/bin/cypress run -s cypress/integration/sample_spec.js --reporter junit",
    "cypress:open": "cypress open"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "cypress": "^3.2.0",
    "mocha": "^5.2.0",
    "mochawesome": "^3.1.1",
    "mochawesome-merge": "^1.0.7",
    "mochawesome-report-generator": "^3.1.5"
  }
}
```

Cypress must be configured to use the "junit" reporter. This can be configured in `cypress.json` file, along with any other options related with the "mocha-junit-formatter" node module.

cypress.json

```
{
  "reporter": "junit",
  "reporterOptions": {
    "mochaFile": "results/test-results.xml",
    "testCaseSwitchClassnameAndName": false
  }
}
```

We then create two tests: a dummy one with an assertion and another one, a bit more complete, that visits a web page and performs some checks against it.

The final check has a bug on its specification, on purpose, just to make the test fail.

cypress/integration/sample_spec.js

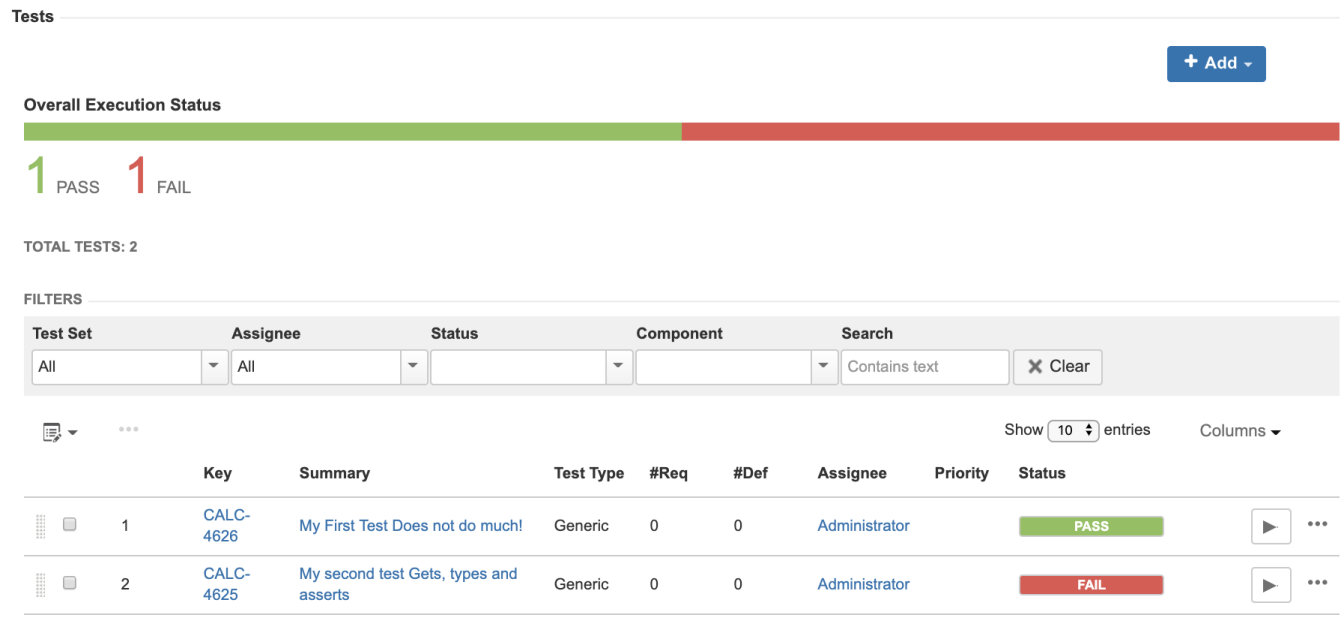
```
describe('My First Test', function() {
  it('Does not do much!', function() {
    expect(true).to.equal(true)
  })
})

describe('My second test', function() {
  it('Gets, types and asserts', function() {
    cy.visit('https://example.cypress.io')

    cy.contains('type').click()

    // Should be on a new URL which includes '/commands/actions'
    cy.url().should('include', '/commands/actions')

    // Get an input, type into it and verify that the value has been updated
    cy.get('.action-email')
      .type('fake@email.com')
      .should('have.value', 'fake@email.comx') // error added on the test itself just to make it fail
  })
})
```

In order to run the tests, we need to execute the following command (or invoke "cypress" directly).

```
npm test
# ./node_modules/cypress/bin/cypress run -s cypress/integration/sample_spec.js --reporter junit
```

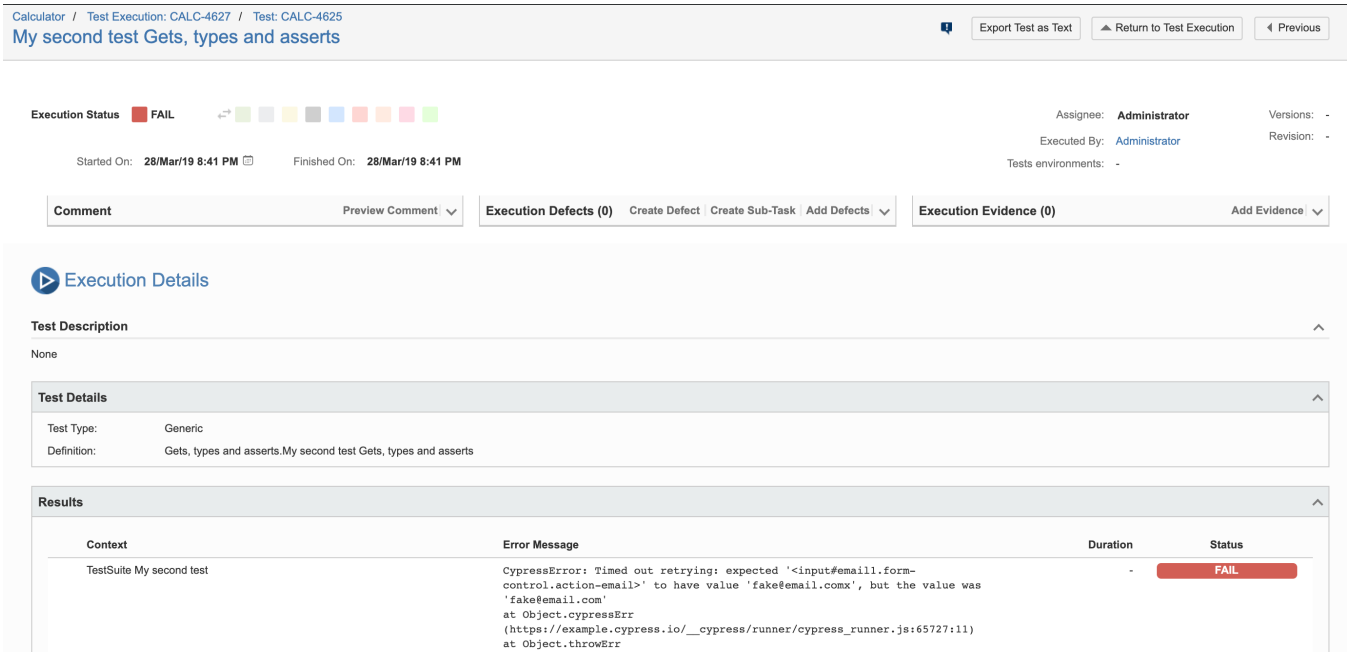
After running the tests and generating the JUnit XML reports (e.g. test-results.xml), they can be imported to Xray (either by the REST API, or by one of the CI plugins Xray has or through the **Import Execution Results** action within the Test Execution).

```
curl -H "Content-Type: multipart/form-data" -u admin:admin -F "file=@results/test-results.xml" http://jiraserver
/rest/raven/1.0/import/execution/junit?projectKey=CALC
```



Each test is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the value of the "it" concatenated with the "describe" element along with the "it" element again. It is possible to configure the behaviour of "mocha-junit-reporter" to behave differentely; for example, setting "testCaseSwitchClassnameAndName" to true on `cypress.json` configuration file, will generate a slight different XML report with different `name` and `classname` attributes on the `testcase` element. Please see Taking advantage of JUnit XML reports for more info on the mapping done based on JUnit XML report.

The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to the concatenation of the test's "describe".



References

- https://www.cypress.io/
- https://docs.cypress.io/guides/getting-started/installing-cypress.html#System-requirements

- https://github.com/michaelleeallen/mocha-junit-reporter