npm init –y

npm i protractor

npm i -g jasmine

protractor conf.js

npm i jasmine

npm install protractor-jasmine2-screenshot-reporter --save-dev

npm i -g jasmine-allure-reporter

npm i jasmine-allure-reporter

npm i -g allure-commandline

allure serve

conf.js

// An example configuration file.

var HtmlScreenshotReporter = require('protractor-jasmine2-screenshot-reporter');

var reporter = new HtmlScreenshotReporter({

dest: 'target/screenshots',

filename: 'my-report.html'

});

exports.config = {

directConnect: true,

// Capabilities to be passed to the webdriver instance.

capabilities: {

'browserName': 'chrome'

},

// Framework to use. Jasmine is recommended.

framework: 'jasmine',

// Spec patterns are relative to the current working directory when

// protractor is called.

specs: ['../tests/calculator.js'],

// Options to be passed to Jasmine.

jasmineNodeOpts: {

defaultTimeoutInterval: 30000

},

// Setup the report before any tests start

beforeLaunch: function () {

return new Promise(function (resolve) {

reporter.beforeLaunch(resolve);

});

},

// Assign the test reporter to each running instance

onPrepare: function () {

var jasmineReporters = require('jasmine-reporters');

jasmine.getEnv().addReporter(new jasmineReporters.JUnitXmlReporter({

consolidateAll: true,

savePath: './',

filePrefix: 'xmlresults'

}));

var fs = require('fs-extra');

fs.emptyDir('screenshots/', function (err) {

console.log(err);

});

jasmine.getEnv().addReporter({

specDone: function(result) {

if (result.status == 'failed') {

browser.getCapabilities().then(function (caps) {

var browserName = caps.get('browserName');

browser.takeScreenshot().then(function (png) {

var stream = fs.createWriteStream('screenshots/' + browserName + '-' + result.fullName+ '.png');

stream.write(new Buffer(png, 'base64'));

stream.end();

});

});

}

}

});

jasmine.getEnv().addReporter(reporter);

var AllureReporter = require('jasmine-allure-reporter');

jasmine.getEnv().addReporter(new AllureReporter({

resultsDir: 'allure-results'

}));

},

// Close the report after all tests finish

afterLaunch: function (exitCode) {

return new Promise(function (resolve) {

reporter.afterLaunch(resolve.bind(this, exitCode));

});

},

//HTMLReport called once tests are finished

onComplete: function () {

var browserName, browserVersion;

var capsPromise = browser.getCapabilities();

capsPromise.then(function (caps) {

browserName = caps.get('browserName');

browserVersion = caps.get('version');

platform = caps.get('platform');

var HTMLReport = require('protractor-html-reporter-2');

testConfig = {

reportTitle: 'Protractor Test Execution Report',

outputPath: './',

outputFilename: 'ProtractorTestReport',

screenshotPath: './screenshots',

testBrowser: browserName,

browserVersion: browserVersion,

modifiedSuiteName: false,

screenshotsOnlyOnFailure: true,

testPlatform: platform

};

new HTMLReport().from('xmlresults.xml', testConfig);

});

}

};

12

npm i protractor-html-reporter-2

 // An example configuration file.

 var HtmlScreenshotReporter = require('protractor-jasmine2-screenshot-reporter');

 var reporter = new HtmlScreenshotReporter({

     dest: 'target/screenshots',

     filename: 'my-report.html'

 });

 exports.config = {

     directConnect: true,

     // Capabilities to be passed to the webdriver instance.

     capabilities: {

         'browserName': 'chrome'

     },

     // Framework to use. Jasmine is recommended.

     framework: 'jasmine',

     // Spec patterns are relative to the current working directory when

     // protractor is called.

     specs: ['../tests/calculator.js'],

     // Options to be passed to Jasmine.

     jasmineNodeOpts: {

         defaultTimeoutInterval: 30000

     },

     // Setup the report before any tests start

     beforeLaunch: function() {

         return new Promise(function(resolve) {

             reporter.beforeLaunch(resolve);

         });

     },

     // Assign the test reporter to each running instance

     onPrepare: function() {

         var jasmineReporters = require('jasmine-reporters');

         jasmine.getEnv().addReporter(new jasmineReporters.JUnitXmlReporter({

             consolidateAll: true,

             savePath: './',

             filePrefix: 'xmlresults'

         }));

         jasmine.getEnv().addReporter(reporter);

         var AllureReporter = require('jasmine-allure-reporter');

         jasmine.getEnv().addReporter(new AllureReporter({

             resultsDir: 'allure-results'

         }));

     },

     // Close the report after all tests finish

     afterLaunch: function(exitCode) {

         return new Promise(function(resolve) {

             reporter.afterLaunch(resolve.bind(this, exitCode));

         });

     },

     onComplete: function() {

         var browserName, browserVersion;

         var capsPromise = browser.getCapabilities();

         capsPromise.then(function(caps) {

             browserName = caps.get('browserName');

             browserVersion = caps.get('version');

             platform = caps.get('platform');

             var HTMLReport = require('protractor-html-reporter-2');

             testConfig = {

                 reportTitle: 'Protractor Test Execution Report',

                 outputPath: './',

                 outputFilename: 'ProtractorTestReport',

                 screenshotPath: './screenshots',

                 testBrowser: browserName,

                 browserVersion: browserVersion,

                 modifiedSuiteName: false,

                 screenshotsOnlyOnFailure: true,

                 testPlatform: platform

             };

             new HTMLReport().from('xmlresults.xml', testConfig);

         });

     }

 }