Use the screenshot reporter to capture screenshots from your Selenium nodes after each executed Protractor test case.

Use meta data stored as JSON along with the screenshots to create extensive and human readable test reports

Usage

The protractor-screenshot-reporter is available via npm:

```
$ npm install protractor-screenshot-reporter --save-dev
```

In your Protractor configuration file, register protractor-screenshot-reporter in Jasmine:

```
var ScreenShotReporter = require('protractor-screenshot-reporter');

exports.config = {
  // your config here ...

  onPrepare: function() {
    // Add a screenshot reporter and store screenshots to `/tmp/screnshots`:
    jasmine.getEnv().addReporter(new ScreenShotReporter({
      baseDirectory: '/tmp/screenshots'
    }));
  }
}
```

Configuration

Base Directory (mandatory)

You have to pass a directory path as parameter when creating a new instance of the screenshot reporter:

```
var reporter = new ScreenShotReporter({
  baseDirectory: '/tmp/screenshots'
});
```

If the given directory does not exists, it is created automatically as soon as a screenshot needs to be stored.

Path Builder (optional)

The function passed as second argument to the constructor is used to build up paths for screenshot files:

```
var path = require('path');

new ScreenShotReporter({
```

```
  baseDirectory: '/tmp/screenshots'
  , pathBuilder: function pathBuilder(spec, descriptions, results, capabilities) {
    // Return '<browser>/<specname>' as path for screenshots:
    // Example: 'firefox/list-should work'.
    return path.join(capabilities.caps_.browserName, descriptions.join('-'));
  }
});
```

If you omit the path builder, a GUID is used by default instead.

Meta Data Builder (optional)

You can modify the contents of the JSON meta data file by passing a function metaDataBuilder function as third constructor parameter:

```
new ScreenShotReporter({
  baseDirectory: '/tmp/screenshots'
  , metaDataBuilder: function metaDataBuilder(spec, descriptions, results, capabilities) {
    // Return the description of the spec and if it has passed or not:
    return {
      description: descriptions.join(' ')
      , passed: results.passed()
    };
  }
});
```

If you omit the meta data builder, the default implementation is used (see https://github.com/swissmanu/protractor-screenshot-reporter/blob/master/index.js#L42).

Screenshots for skipped test cases (optional)

Since version 0.0.3, you can define if you want capture screenshots from skipped test cases using the takeScreenShotsForSkippedSpecs option:

```
new ScreenShotReporter({
  baseDirectory: '/tmp/screenshots'
  , takeScreenShotsForSkippedSpecs: true
});
```

Default is false.

Screenshots only for failed test cases (optional)

Also you can define if you want capture screenshots only from failed test cases using the takeScreenShotsOnlyForFailedSpecs: option:

```
new ScreenShotReporter({
  baseDirectory: '/tmp/screenshots'
  , takeScreenShotsOnlyForFailedSpecs: true
```

});

Default is false.

Postprocess Meta Data

A screenshot is saved as PNG image. Along with it, a JSON file with a matching filename is created.

By default, the JSON file contains the test outcome and further information about the Selenium machine which executed the test case:


```
{
  "description":"index.html header should fail",
  "passed":false,
  "os":"LINUX",
  "browser":{
    "name":"firefox",
    "version":"26.0"
  },
  "screenshot":"004200db-0032-005a-00f3-0072008c001c.png",
  "message":"TypeError: Cannot read property 'be' of undefined",
  "trace":"TypeError: Cannot read property 'be' of undefined\n    at null.<anonymous> etc."
}
```


By parsing these JSON files in a preprocessing step, you can generate a human readable test report.

Postprocessing is not a functionality of protractor-screenshot-reporter.

Use Grunt or similar tools to integrate it into your test process.

****