How to Create a Cucumber Reporter using multiple-cucumber-html-reporter

1. Introduction to Cucumber Reporters

- Cucumber provides a default reporter for basic information such as pass/fail.
- For advanced dashboards, system metadata, and a cleaner UI, use the `multiple-cucumber-html-reporter` package.

---

2. Default Cucumber Report

In your `cucumber.js` configuration file, add:

```
module.exports = {
  default: {
    format: [
      "html:test-results/cucumber-report.html",
      "json:test-results/cucumber-report.json",
      "junit:test-results/cucumber-report.xml"
    ]
  }
};
```

- Provides basic status like pass/fail.
- Does not include dashboard view or system information.

---

3. Why Use `multiple-cucumber-html-reporter`?

This third-party reporter provides:

- A visual dashboard UI
- System info (who executed the test, project name, release version)
- Optional progress bar in console

To enable the progress bar in the console, update `cucumber.js` like this:

```
format: [
  "json:test-results/cucumber-report.json",
  "progress-bar"
]
```

---

4. Generate JSON Results

Update `cucumber.js`:

```
format: [
  "json:test-results/cucumber-report.json"
]
```

This step creates the `cucumber-report.json` file after the test run.

---

5. Install the Reporter

Install the required package via npm:

```
npm install multiple-cucumber-html-reporter --save-dev
```

- Confirm it's listed under `devDependencies` in your `package.json`.

---

6. Create a Report Generator Script

Step 1: Create a helper folder

```
mkdir src/helper
```

Step 2: Create `report.ts` inside `helper/`

```
const report = require("multiple-cucumber-html-reporter");

report.generate({
  jsonDir: 'test-results/cucumber-json',
  reportPath: 'test-results/cucumber-report',
  storeScreenshots: true,
  displayDuration: true,
  openReportInBrowser: true,
  screenshotsDirectory: 'test-results/screenshots/',
  pageTitle: 'Cucumber Test Report',
  reportName: 'Cucumber Test Execution Report',
  customData: {
    title: 'Test Execution Information',
    data: [
      { label: 'Project', value: 'Playwright Cucumber Tests' },
      { label: 'Version', value: '1.0.0' },
      { label: 'Execution Date', value: new Date().toLocaleString() }
    ]
  }
});
```

---

7. Link in `package.json`

Update the `scripts` section:

```
"scripts": {
  "test": "cucumber-js test",
  "posttest": "npx ts-node src/helper/report.ts"
}
```

- `posttest` runs automatically after `test`, generating the HTML report.

---

8. Handling Test Failures Gracefully

To ensure the report is generated even if tests fail:

```
"scripts": {
  "test": "cucumber-js --config=config/cucumber.js || exit 0",
  "posttest": "npx ts-node src/helper/report/report.ts"
}
```

The `|| exit 0` ensures `posttest` runs regardless of test outcomes.

Execution Flow

1. Run tests using `npm run test`.
2. JSON report is generated at `test-results/cucumber-report.json`.
3. The `posttest` script triggers `report.ts`.
4. The HTML report is created and can be viewed at: `test-results/cucumber-report/index.html`.

Created By: Yogesh Pandian