Playwright Screenshots, Videos and Tracing (Trace Viewer)

Screenshots in Playwright

1. Capture a Screenshot

To take a screenshot of the current browser view:

```
await page.screenshot({ path: 'screenshot.png' });
```

2. Full Page Screenshot

Captures the entire scrollable page, as if it fits in one tall screen:

```
await page.screenshot({ path: 'screenshot.png', fullPage: true });
```

3. Capture an Element Screenshot

Takes a screenshot of a specific element on the page:

```
await page.locator('.header').screenshot({ path: 'screenshot.png' });
```

4. Screenshot Settings in playwright.config.ts

You can configure when screenshots are captured by setting options in the config file:

```
export default defineConfig({
  use: {
    screenshot: 'only-on-failure', // Options: 'on', 'off', 'only-on-failure', 'on-first-failure'
  },
});
```

Screenshot Options:

- 'on': Always take a screenshot after each test.
- 'off': Do not capture screenshots (default).
- 'only-on-failure': Capture screenshots only when a test fails.
- 'on-first-failure': Capture screenshot only on the first failure of a test.

Note:

- Screenshots are saved inside the **test-results** folder.
- Screenshots are linked in the Playwright HTML test reports.

Video Recording in Playwright

Configure Video Capture in **playwright.config.ts file.**

```
export default defineConfig({
  use: {
    video: 'on-first-retry', // Options: 'on', 'off', 'retain-on-failure', 'on-first-retry'
  },
});
```

Video Options:

- 'off': No video recording.
- 'on': Record video for all tests.
- 'retain-on-failure': Keep videos only for failed tests.
- 'on-first-retry': Record video only when a test is retried for the first time.

Note: Video files are saved in the **test-results** folder. And they embed with HTML report.

Tracing and Trace Viewer

Playwright Tracing records actions, snapshots, and events during a test run. This trace can be opened later in a GUI tool called **Trace Viewer**.

Ways to Enable Tracing:

1. In playwright.config.ts

```
export default defineConfig({
  use: {
    trace: 'on', // Options: 'off', 'on', 'retain-on-failure', 'on-first-retry'
  },
});
```

2. Using CLI

```
npx playwright test tests/example.spec.ts --headed --trace on
```

3. Programmatically in Code

```
test('tracing test', async ({ page, context }) => {
  await context.tracing.start({ screenshots: true, snapshots: true });
    // Test actions go here
  await context.tracing.stop({ path: 'trace.zip' });
});
```

Note: The trace.zip file generated is not automatically shown in the HTML report.

Viewing Trace Files:

There are **3 ways** to view trace files:

1. From HTML Report

- Run:

```
npx playwright show-report
```

- In the report, click on the **trace.zip** link to open Trace Viewer.

2. Using Command Line

```
npx playwright show-trace trace.zip
```

3. Using Online Viewer

- Open: https://trace.playwright.dev/
- Drag and drop the trace.zip file to view it.

Troubleshooting: Report Port in Use Error

If you encounter:

```
Error: listen EADDRINUSE: address already in use ::1:9323
```

Solution 1: Kill the process using the port

```
netstat -ano | findstr :9323

taskkill /PID <PID> /F
```

Solution 2: Use a different port

```
npx playwright show-report --port=9324
```

Flaky Tests and Retries

What are Flaky Tests?

A **flaky test** is a test that sometimes passes and sometimes fails, even when there is no change in the code. These failures are usually caused by things like:

- Slow network or server response
- Delayed UI updates
- Timing issues or animations

Such tests are **not reliable** and can make it hard to trust your test results.

How Playwright Helps: Retries

To handle flaky tests, **Playwright allows you to retry tests** that fail. If a test fails, Playwright can **automatically run it again**, up to a set number of times.

Example Scenarios:

1. ✅ **Test Passed** → No retry needed
2. ❌ **Test Failed** → Retry → ❌ Still Failed
3. ❌ **Test Failed** → Retry → ✅ Passed → **This is called a flaky test**

How to Use Retries in Playwright

1. Configure in playwright.config.ts file:

You can set how many times to retry a failed test like this:

```
export default defineConfig({
  retries: 3, // This will retry a failed test up to 3 times
});
```

2. Or Use CLI (Command Line Interface):

You can also set retries while running your tests from the terminal:

# Run all tests with 3 retry attempts

```
npx playwright test --retries=3
```

# Run a specific test file with retries

```
npx playwright test tests/flakytest.spec.ts --retries=3
```