Visual comparisons

Introduction

Playwright Test includes the ability to produce and visually compare screenshots using `await expect(page).toHaveScreenshot()`. On first execution, Playwright test will generate reference screenshots. Subsequent runs will compare against the reference.

example.spec.ts

```
import { test, expect } from '@playwright/test';

test('example test', async ({ page }) => {
  await page.goto('https://playwright.dev');
  await expect(page).toHaveScreenshot();
});
```

Generating screenshots

When you run above for the first time, test runner will say:

```
Error: A snapshot doesn't exist at example.spec.ts-snapshots/example-test-1-chromium-darwin.png, writing actual.
```

That's because there was no golden file yet. This method took a bunch of screenshots until two consecutive screenshots matched, and saved the last screenshot to file system. It is now ready to be added to the repository.

The name of the folder with the golden expectations starts with the name of your test file:

```
drwxr-xr-x  5 user  group  160 Jun  4 11:46 .
drwxr-xr-x  6 user  group  192 Jun  4 11:45 ..
-rw-r--r--  1 user  group  231 Jun  4 11:16 example.spec.ts
drwxr-xr-x  3 user  group   96 Jun  4 11:46 example.spec.ts-snapshots
```

The snapshot name `example-test-1-chromium-darwin.png` consists of a few parts:

- `example-test-1.png` - an auto-generated name of the snapshot. Alternatively you can specify snapshot name as the first argument of the `toHaveScreenshot()` method:

  ```
  await expect(page).toHaveScreenshot('landing.png');
  ```

- `chromium-darwin` - the browser name and the platform. Screenshots differ between browsers and platforms due to different rendering, fonts and more, so you will need different snapshots for them. If you use multiple projects in your configuration file, project name will be used instead of `chromium`.

The snapshot name and path can be configured with `snapshotPathTemplate` in the playwright config.

Updating screenshots

Sometimes you need to update the reference screenshot, for example when the page has changed. Do this with the `--update-snapshots` flag.

```
npx playwright test --update-snapshots
```

> Note that `snapshotName` also accepts an array of path segments to the snapshot file such as `expect().toHaveScreenshot(['relative', 'path', 'to', 'snapshot.png'])`. However, this path must stay within the snapshots directory for each test file (i.e. `a.spec.js-snapshots`), otherwise it will throw.

Options

maxDiffPixels

Playwright Test uses the pixelmatch library. You can pass various options to modify its behavior:

example.spec.ts
```ts
import { test, expect } from '@playwright/test';

test('example test', async ({ page }) => {
  await page.goto('https://playwright.dev');
  await expect(page).toHaveScreenshot({ maxDiffPixels: 100 });
});
```

If you'd like to share the default value among all the tests in the project, you can specify it in the playwright config, either globally or per project:

playwright.config.ts
```ts
import { defineConfig } from '@playwright/test';
export default defineConfig({
  expect: {
    toHaveScreenshot: { maxDiffPixels: 100 },
  },
});
```

stylePath

You can apply a custom stylesheet to your page while taking screenshot. This allows filtering out dynamic or volatile elements, hence improving the screenshot determinism.

screenshot.css

```
iframe {
  visibility: hidden;
}
```

example.spec.ts

```
import { test, expect } from '@playwright/test';

test('example test', async ({ page }) => {
  await page.goto('https://playwright.dev');
  await expect(page).toHaveScreenshot({ stylePath: path.join(__dirname,
'screenshot.css') });
});
```

If you'd like to share the default value among all the tests in the project, you can specify it in the playwright config, either globally or per project:

playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  expect: {
    toHaveScreenshot: {
      stylePath: './screenshot.css'
    },
  },
});
```

Non-image snapshots

Apart from screenshots, you can use `expect(value).toMatchSnapshot(snapshotName)` to compare text or arbitrary binary data. Playwright Test auto-detects the content type and uses the appropriate comparison algorithm.

Here we compare text content against the reference.

example.spec.ts

```
import { test, expect } from '@playwright/test';

test('example test', async ({ page }) => {
  await page.goto('https://playwright.dev');
  expect(await
page.textContent('.hero__title')).toMatchSnapshot('hero.txt');
});
```

Snapshots are stored next to the test file, in a separate directory. For example, `my.spec.ts` file will produce and store snapshots in the `my.spec.ts-snapshots` directory. You should commit this directory to your version control (e.g. `git`), and review any changes to it.

Previous
Screenshots

Next
Test generator

Learn

Community

More

Copyright © 2024 Microsoft