Retries

**Introduction**

Test retries are an effective method to re-run tests automatically when they fail. This feature is particularly useful when dealing with flaky tests that fail intermittently. The retry mechanism is configured in the Playwright configuration file.

**Failure Handling**

Playwright Test operates by running tests in separate worker processes. Each worker process is a standalone OS process that starts its own browser and is managed independently by the test runner. All workers operate in identical environments.

Here is an example demonstrating this setup:

ts

Copy code

import { test } from '@playwright/test';

test.describe('suite', () => {

test.beforeAll(async () => { /\* ... \*/ });

test('first good', async ({ page }) => { /\* ... \*/ });

test('second flaky', async ({ page }) => { /\* ... \*/ });

test('third good', async ({ page }) => { /\* ... \*/ });

test.afterAll(async () => { /\* ... \*/ });

});

In normal circumstances, where all tests pass, they are executed in sequence within the same worker process.

**Scenario**:

* **Worker process starts**
* beforeAll hook executes
* first good passes
* second flaky passes
* third good passes
* afterAll hook runs

However, when a test fails, Playwright Test discards the entire worker process, including the browser, and spawns a new one. The remaining tests resume in the new process.

**Scenario with Failure**:

* **Worker process #1** starts
* beforeAll hook executes
* first good passes
* second flaky fails
* afterAll hook runs
* **Worker process #2** starts
* beforeAll hook executes again
* third good passes
* afterAll hook runs

If retries are enabled, the second worker process begins by retrying the failed test, then continues with the rest of the tests.

**Scenario with Retries**:

* **Worker process #1** starts
* beforeAll hook executes
* first good passes
* second flaky fails
* afterAll hook runs
* **Worker process #2** starts
* beforeAll hook runs again
* second flaky is retried and passes
* third good passes
* afterAll hook runs

This approach works well for independent tests and ensures that a failing test won't affect other tests.

**Enabling Retries**

Playwright supports test retries, which means failing tests are retried automatically. You can specify the number of retries. By default, retries are disabled.

Run the command below to enable retries with three attempts:

bash

Copy code

npx playwright test --retries=3

Alternatively, you can configure retries in the Playwright configuration file:

ts

Copy code

// playwright.config.ts

import { defineConfig } from '@playwright/test';

export default defineConfig({

retries: 3, // Retry failing tests 3 times

});

Playwright Test categorizes tests as follows:

* **Passed**: Tests that pass on the first run.
* **Flaky**: Tests that fail initially but pass on retry.
* **Failed**: Tests that fail both the initial run and all retries.

**Example Output with Retries**

Running a set of three tests with retries enabled:

plaintext

Copy code

Running 3 tests using 1 worker

✓ example.spec.ts:4:2 › first passes (438ms)

x example.spec.ts:5:2 › second flaky (691ms)

✓ example.spec.ts:5:2 › second flaky (522ms)

✓ example.spec.ts:6:2 › third passes (932ms)

1 flaky

example.spec.ts:5:2 › second flaky

2 passed (4s)

**Detecting Retries at Runtime**

Playwright provides testInfo.retry to detect when a test is being retried. You can use this to perform specific actions, such as clearing caches, before a retry:

ts

Copy code

import { test } from '@playwright/test';

test('my test', async ({ page }, testInfo) => {

if (testInfo.retry) {

await cleanSomeCachesOnTheServer();

}

// Test logic here...

});

**Configuring Retries for Specific Tests**

You can configure retries for specific tests or test groups using test.describe.configure():

ts

Copy code

import { test } from '@playwright/test';

test.describe(() => {

// Set 2 retry attempts for this group of tests

test.describe.configure({ retries: 2 });

test('test 1', async ({ page }) => { /\* ... \*/ });

test('test 2', async ({ page }) => { /\* ... \*/ });

});

**Serial Mode**

The test.describe.serial() function ensures that dependent tests run together in order, and if one test fails, all subsequent tests are skipped. In this mode, all tests in the group are retried together.

ts

Copy code

import { test } from '@playwright/test';

test.describe.configure({ mode: 'serial' });

test.beforeAll(async () => { /\* ... \*/ });

test('first good', async ({ page }) => { /\* ... \*/ });

test('second flaky', async ({ page }) => { /\* ... \*/ });

test('third good', async ({ page }) => { /\* ... \*/ });

If retries are enabled, all tests are retried together, as shown below:

plaintext

Copy code

Worker process #1:

beforeAll hook runs

first good passes

second flaky fails

third good is skipped

Worker process #2:

beforeAll hook runs again

first good passes again

second flaky passes

third good passes

**Isolated Tests vs. Serial Mode**

While serial mode is useful for dependent tests, it is generally better to isolate your tests so they can be retried independently for better performance and reliability.

**Reusing a Single Page Between Tests**

Although Playwright Test isolates the Page object by default, you can share a single Page instance across multiple tests by creating it in the beforeAll() and closing it in afterAll().

ts

Copy code

const { test } = require('@playwright/test');

test.describe.configure({ mode: 'serial' });

let page;

test.beforeAll(async ({ browser }) => {

page = await browser.newPage();

});

test.afterAll(async () => {

await page.close();

});

test('runs first', async () => {

await page.goto('https://playwright.dev/');

});

test('runs second', async () => {

await page.getByText('Get Started').click();

});