2.Network

Playwright allows users to monitor and modify browser network traffic, including HTTP and HTTPS. This feature enables tracking, modifying, and handling of all requests, including XHRs and fetch requests.

* **Key Benefits:**
  + Monitor API traffic and mock responses.
  + Block, modify, or abort network requests dynamically.

**Example:**

typescript

Copy code

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

test.beforeEach(async ({ context }) => {

// Block any CSS requests before each test

await context.route(/\.css$/, route => route.abort());

});

test('page without CSS', async ({ page }) => {

await page.goto('https://example.com');

});

**10.2 Mocking APIs**

Playwright can mock network requests without hitting the actual API using various methods, such as intercepting API calls or using HAR files.

**10.2.1 Mocking API Requests**

* You can intercept API requests and return a mocked response, allowing for tests that don't rely on the live API.

**Example:**

typescript

Copy code

test('mock API response', async ({ page }) => {

await page.route('\*\*/api/v1/fruits', route => {

const mockResponse = [{ name: 'Mango', id: 22 }];

route.fulfill({ json: mockResponse });

});

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

});

**10.2.2 Using HAR Files for Network Mocking**

* HAR (HTTP Archive) files can record network requests and be replayed to mock responses.

**Example:**

typescript

Copy code

test('use HAR for mocking', async ({ page }) => {

await page.routeFromHAR('./mockData.har', { url: '\*\*/api/v1/fruits', update: false });

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

});

**10.3 HTTP Authentication**

Playwright supports HTTP Authentication, allowing tests to simulate login by passing credentials.

* **Usage:** Specify username and password in the configuration file.

**Example:**

typescript

Copy code

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

export default defineConfig({

use: {

httpCredentials: {

username: 'user123',

password: 'pass123',

}

}

});

**10.4 HTTP Proxy Configuration**

Playwright allows setting up HTTP or SOCKS proxy for the browser globally or per browser context.

**10.4.1 Global Proxy Configuration**

* A proxy server can be globally configured to reroute all network requests.

**Example:**

typescript

Copy code

export default defineConfig({

use: {

proxy: {

server: 'http://proxyserver.com:3128',

username: 'proxyuser',

password: 'proxypass'

}

}

});

**10.4.2 Context-Specific Proxy**

* Each browser context can have its own proxy configuration.

**Example:**

typescript

Copy code

test('context with proxy', async ({ browser }) => {

const context = await browser.newContext({

proxy: {

server: 'http://myproxy.com:3128',

}

});

});

**10.5 Network Events**

Playwright provides ways to monitor and log network requests and responses.

* **Request & Response Events:** You can capture requests/responses made by the page.

**Example:**

typescript

Copy code

page.on('request', request => console.log('Request:', request.url()));

page.on('response', response => console.log('Response:', response.url()));

**10.5.1 Waiting for Network Responses**

* Wait for specific network responses during actions such as button clicks.

**Example:**

typescript

Copy code

const response = await page.waitForResponse('\*\*/api/fetch\_data');

await page.getByText('Submit').click();

**10.6 Handling and Modifying Requests**

Playwright allows you to modify or continue requests with altered payloads, headers, or methods.

**10.6.1 Modify Headers**

* You can modify outgoing headers, such as removing or adding custom headers.

**Example:**

typescript

Copy code

await page.route('\*\*/\*', route => {

const headers = route.request().headers();

delete headers['X-Custom-Header'];

route.continue({ headers });

});

**10.6.2 Abort Requests**

* Certain requests can be aborted dynamically based on patterns or resource types.

**Example:**

typescript

Copy code

await page.route('\*\*/\*.{png,jpg}', route => route.abort());

**10.7 WebSockets Handling**

Playwright supports WebSocket traffic inspection, allowing users to log or monitor WebSocket events.

**Example:**

typescript

Copy code

page.on('websocket', ws => {

ws.on('framesent', data => console.log('Frame sent:', data));

ws.on('framereceived', data => console.log('Frame received:', data));

});

**10.8 Missing Network Events and Service Workers**

Network events might be missing if Service Workers interfere. Disable Service Workers to capture requests using Playwright's browserContext.route().

**Note:** If using tools like MSW (Mock Service Worker), it may take over the network requests and prevent Playwright from listening to them.