WebView2

Introduction

The following will explain how to use Playwright with Microsoft Edge WebView2. WebView2 is a WinForms control, which will use Microsoft Edge under the hood to render web content. It is a part of the Microsoft Edge browser and is available on Windows 10 and Windows 11. Playwright can be used to automate WebView2 applications and can be used to test web content in WebView2. For connecting to WebView2, Playwright uses browserType.connectOverCDP() which connects to it via the Chrome DevTools Protocol (CDP).

Overview

A WebView2 control can be instructed to listen to incoming CDP connections by setting either the `WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS` environment variable with `--remote-debugging-port=9222` or calling EnsureCoreWebView2Async with the `--remote-debugging-port=9222` argument. This will start the WebView2 process with the Chrome DevTools Protocol enabled which allows the automation by Playwright. 9222 is an example port in this case, but any other unused port can be used as well.

```
await this.webView.EnsureCoreWebView2Async(await
CoreWebView2Environment.CreateAsync(null, null, new
CoreWebView2EnvironmentOptions()
{
  AdditionalBrowserArguments = "--remote-debugging-port=9222",
})).ConfigureAwait(false);
```

Once your application with the WebView2 control is running, you can connect to it via Playwright:

```
const browser = await
playwright.chromium.connectOverCDP('http://localhost:9222');
const context = browser.contexts()[0];
const page = context.pages()[0];
```

To ensure that the WebView2 control is ready, you can wait for the `CoreWebView2InitializationCompleted` event:

```
this.webView.CoreWebView2InitializationCompleted += (_, e) =>
{
```

```
    if (e.IsSuccess)
    {
        Console.WriteLine("WebView2 initialized");
    }
};
```

Writing and running tests

By default, the WebView2 control will use the same user data directory for all instances. This means that if you run multiple tests in parallel, they will interfere with each other. To avoid this, you should set the `WEBVIEW2_USER_DATA_FOLDER` environment variable (or use WebView2.EnsureCoreWebView2Async Method) to a different folder for each test. This will make sure that each test runs in its own user data directory.

Using the following, Playwright will run your WebView2 application as a sub-process, assign a unique user data directory to it and provide the Page instance to your test:

webView2Test.ts

```
import { test as base } from '@playwright/test';
import fs from 'fs';
import os from 'os';
import path from 'path';
import childProcess from 'child_process';

const EXECUTABLE_PATH = path.join(
    __dirname,
    '../../webview2-app/bin/Debug/net8.0-windows/webview2.exe',
);

export const test = base.extend({
  browser: async ({ playwright }, use, testInfo) => {
    const cdpPort = 10000 + testInfo.workerIndex;
    // Make sure that the executable exists and is executable
    fs.accessSync(EXECUTABLE_PATH, fs.constants.X_OK);
    const userDataDir = path.join(
        fs.realpathSync.native(os.tmpdir()),
        `playwright-webview2-tests/user-data-dir-${testInfo.workerIndex}`,
    );
    const webView2Process = childProcess.spawn(EXECUTABLE_PATH, [], {
      shell: true,
      env: {
        ...process.env,
        WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS: `--remote-debugging-
port=${cdpPort}`,
        WEBVIEW2_USER_DATA_FOLDER: userDataDir,
      }
    });
    await new Promise<void>(resolve => webView2Process.stdout.on('data',
data => {
      if (data.toString().includes('WebView2 initialized'))
        resolve();
    }));
    const browser = await
playwright.chromium.connectOverCDP(`http://127.0.0.1:${cdpPort}`);
    await use(browser);
    await browser.close();
    childProcess.execSync(`taskkill /pid ${webView2Process.pid} /T /F`);
    fs.rmdirSync(userDataDir, { recursive: true });
```

```
  },
  context: async ({ browser }, use) => {
    const context = browser.contexts()[0];
    await use(context);
  },
  page: async ({ context }, use) => {
    const page = context.pages()[0];
    await use(page);
  },
});

export { expect } from '@playwright/test';
```

example.spec.ts

```
import { test, expect } from './webView2Test';

test('test WebView2', async ({ page }) => {
  await page.goto('https://playwright.dev');
  const getStarted = page.getByText('Get Started');
  await expect(getStarted).toBeVisible();
});
```

Debugging

Inside your webview2 control, you can just right-click to open the context menu and select "Inspect" to open the DevTools or press `F12`. You can also use the WebView2.CoreWebView2.OpenDevToolsWindow method to open the DevTools programmatically.

For debugging tests, see the Playwright Debugging guide.

Previous
Videos

Next
Migrating from Protractor

- Introduction
- Overview
- Writing and running tests
- Debugging

Learn

- Getting started
- Playwright Training
- Learn Videos
- Feature Videos

Community

- Stack Overflow

- Discord
- Twitter
- LinkedIn

More

- GitHub
- YouTube
- Blog
- Ambassadors