# 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 browser_type.connect_over_cdp() 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:

**Sync** Async

```
browser = playwright.chromium.connect_over_cdp("http://localhost:9222")
context = browser.contexts[0]
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:

conftest.py

```
import os
import socket
import tempfile
import pytest
from pathlib import Path
from playwright.sync_api import Playwright, Browser, BrowserContext
import subprocess

EXECUTABLE_PATH = (
    Path(__file__).parent
    / ".."
    / "webview2-app"
    / "bin"
    / "Debug"
    / "net6.0-windows"
    / "webview2.exe"
)
```

```
@pytest.fixture(scope="session")
def data_dir():
    with tempfile.TemporaryDirectory(
        prefix="playwright-webview2-tests", ignore_cleanup_errors=True
    ) as tmpdirname:
        yield tmpdirname


@pytest.fixture(scope="session")
def webview2_process_cdp_port(data_dir: str):
    cdp_port = _find_free_port()
    process = subprocess.Popen(
        [EXECUTABLE_PATH],
        env={
            **dict(os.environ),
            "WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS": f"--remote-debugging-port=
{cdp_port}",
            "WEBVIEW2_USER_DATA_FOLDER": data_dir,
        },
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT,
        universal_newlines=True,
    )
    while True:
        line = process.stdout.readline()
        if "WebView2 initialized" in line:
            break
    yield cdp_port
    process.terminate()


@pytest.fixture(scope="session")
def browser(playwright: Playwright, webview2_process_cdp_port: int):
    browser = playwright.chromium.connect_over_cdp(
        f"http://127.0.0.1:{webview2_process_cdp_port}"
    )
    yield browser


@pytest.fixture(scope="function")
def context(browser: Browser):
    context = browser.contexts[0]
    yield context


@pytest.fixture(scope="function")
```

```python
def page(context: BrowserContext):
    page = context.pages[0]
    yield page


def _find_free_port(port=9000, max_port=65535):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    while port <= max_port:
        try:
            sock.bind(("", port))
            sock.close()
            return port
        except OSError:
            port += 1
    raise IOError("no free ports")
```

test_webview2.py

```python
from playwright.sync_api import Page, expect


def test_webview2(page: Page):
    page.goto("https://playwright.dev")
    get_started = page.get_by_text("Get Started")
    expect(get_started).to_be_visible()
```

## 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.