**Playwright Setup and Testing for Blazor Web Applications**

Contents

[1. Installation and Setup 1](#_Toc178692919)

[Node.js Requirement 1](#_Toc178692920)

[**Initializing Playwright** 2](#_Toc178692921)

[**Project Structure** 2](#_Toc178692922)

[**2. Playwright Management and Updates** 2](#_Toc178692923)

[**Starting Playwright (New Project Setup)** 2](#_Toc178692924)

[**Updating Playwright** 2](#_Toc178692925)

[**Installing Browser Dependencies** 2](#_Toc178692926)

[**Checking Playwright Version** 3](#_Toc178692927)

[**4.** **Running Tests** 3](#_Toc178692928)

[**Running All Tests** 3](#_Toc178692929)

[**Running a Specific Test** 4](#_Toc178692930)

[**Viewing HTML Reports** 4](#_Toc178692931)

[**Interactive Debugging with UI** 4](#_Toc178692932)

[**4. Running Tests for Specific Browsers** 4](#_Toc178692933)

[**WebKit (Safari Engine)** 4](#_Toc178692934)

[**Firefox** 4](#_Toc178692935)

[**Chromium (Chrome/Edge)** 4](#_Toc178692936)

[**5. Additional Notes and Useful Commands** 5](#_Toc178692937)

[**Running Tests in Headed Mode** 5](#_Toc178692938)

[**Using Test Filters** 5](#_Toc178692939)

[**Debugging with Breakpoints** 5](#_Toc178692940)

[**Taking Screenshots for Visual Testing** 5](#_Toc178692941)

# 1. Installation and Setup

### Node.js Requirement

* Ensure you have the latest version of **Node.js** installed before starting with Playwright.
* Download it from [Node.js official website](https://nodejs.org/).
* To check if Node.js is installed or verify the version, run the following command in your terminal:

node -v

**Initializing Playwright**

1. Go to your Blazor project’s main folder.
2. Run the following command to initialize Playwright:

npm init playwright@latest

1. This command will:
   * Install Playwright dependencies.
   * Set up a **tests** folder containing .spec.js files (test scripts).
   * Download necessary browser binaries (Chromium, Firefox, WebKit).

**Project Structure**

* Once Playwright is initialized, the default folder structure will look like this:

tests/

├── example.spec.js (\*Sample test file generated by Playwright\*)

└── <your custom test files>

**2. Playwright Management and Updates**

**Starting Playwright (New Project Setup)**

* When setting up a new project or integrating Playwright into an existing Blazor project, initialize Playwright with:

npm init playwright@latest

**Updating Playwright**

* To update Playwright to the latest version, run:

npm install -D @playwright/test@latest

**Installing Browser Dependencies**

* If Playwright requires additional browser dependencies (e.g., for CI environments), install them using:

npx playwright install --with-deps

**Checking Playwright Version**

* To verify the current version of Playwright installed, run:

npx playwright --version

1. **Running Tests**

A Playwright test script typically follows this structure:

* **Launching a browser**.
* **Navigating to a webpage**.
* **Interacting with UI elements**.
* **Verifying expected behaviour** (using assertions).

**Running Recorder**

* **PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp>** cd C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp
* **>>**
* **PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp>** npm init -y
* **PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp>** npm install playwright --save-dev
* **PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp**> npx playwright install
* **>>**
* **Run code in different terminal**(dotnet run)
* **PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp>** npx playwright codegen <http://localhost:5128> (running url)
* Make sure its recording, once finished copy the code make a spec.js((only difference to typescript should be **Run(**npm install ts-node @types/node --save-dev) and the file being saved as spec.ts) file paste the code into it fix the structure. **Run the code using** “npx playwright test –ui”**(better ui),** **or** “npx playwright test”**(answers)**

**Running All Tests**

* To run all tests across multiple browsers (Chromium, Firefox, WebKit), use:

npx playwright test

**Running a Specific Test**

* To run a specific test file (e.g., weather.spec.js), run:

npx playwright test weather.spec.js

**Viewing HTML Reports**

* After running tests, Playwright generates a detailed **HTML report**. To view the report, use:

npx playwright show-report

**Interactive Debugging with UI**

* For an more interactive test/running experience with real-time updates, use:

npx playwright test --ui

* + This will open a UI where you can browse test results, inspect errors, and re-run tests.

**4. Running Tests for Specific Browsers**

Playwright supports testing across multiple browsers, which is essential for cross-browser testing. Below are commands to run tests on specific browsers.

**WebKit (Safari Engine)**

* To run tests on the **WebKit** engine (Safari), use:

npx playwright test --project=webkit

**Firefox**

* To run tests in **Firefox**, use:

npx playwright test --project=firefox

**Chromium (Chrome/Edge)**

* To run tests on the **Chromium** engine (Chrome, Edge), use:

npx playwright test --project=chromium

**5. Additional Notes and Useful Commands**

**Running Tests in Headed Mode**

* Playwright runs in headless mode (no browser UI) by default. If you want to run it with the browser UI visible for debugging purposes, use:

npx playwright test --headed

**Using Test Filters**

* Playwright allows you to filter and run specific tests based on keywords. To run tests that match a particular keyword, use:

npx playwright test --grep "Login"

**Debugging with Breakpoints**

* You can pause test execution at any point using the following command in your test script:

await page.pause();

* + This opens the browser and pauses the test, allowing you to manually inspect the state.

**Taking Screenshots for Visual Testing**

* You can capture screenshots of specific elements or entire pages during your tests:

await page.screenshot({ path: 'example.png' });

\*Screenshots are useful for visual regression testing, ensuring that the UI does not change unexpectedly between releases.\*

PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp> cd C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp

>>

PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp> npm init -y

PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp> npm install playwright --save-dev

PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp> npx playwright install

>>

Run code in different terminal(dotnet run)

PS C:\Users\Boyd\Documents\GitHub\playwrite\BlazorApp> npx playwright codegen <http://localhost:5128>

Make sure its recording, once finished copy the code make a spec.js file past the code into it fix fix the structure. Run the code using “npx playwright test –ui”(better ui), or “npx playwright test”(answers)

**Cs Running**

Install-Package NUnit

Install-Package NUnit3TestAdapter

Install-Package Microsoft.Playwright.NUnit

Change Test Runner to NUnit (in the record window with the test code)

Recording is the same

Copy the code, Save into .cs file

Right click and run Debug Tests