

.NET Conf China
2022

Playwright探索

骆姜斌

智鹏瑞尔软件 (SSW China) – 解决方案架构师

FireUG技术社区组织者之一



B站



抖音



公众号



微信群



目录

什么是Playwright

Playwright VS Selenium

如何使用Playwright

常见问题解答

Demo



.NET Conf China

什么是Playwright ?



什么是Playwright

➤ 微软开源的自动化的Web测试框架

- 部分团队成员曾属于Puppeteer团队
- Puppeteer - PuppeteerSharp
 - Playwright - <https://github.com/microsoft/playwright>
 - Playwright for .NET - <https://github.com/microsoft/playwright-dotnet>
 - Playwright for Java – <https://github.com/microsoft/playwright-java>
 - Playwright for Python - <https://github.com/microsoft/playwright-python>

➤ 主流浏览器支持

- Chromium
- Firefox
- WebKit

➤ 项目活跃度

- 48小时内响应Issue
- 已解决 7000+ Issue
- 每月多次发布



附带的工具

- Codegen

- 脚本录制工具

- Playwright Inspector

- 代码开发和调试的GUI工具

- **Trace Viewer**

- 查看脚本执行过程的GUI工具
- 包含页面截图, 脚本执行以及执行前后的相关信息



Playwright优点

- 一键安装, 自动下载浏览器
- Headless模式下允许录制视频
- 可以根据需要干涉修改请求
- 模拟各种设备
- 代码生成
- 便捷的调试方式



Playwright VS Selenium



开发语言



Playwright

C#

JavaScript

Java

Python

TypeScript

Selenium

C#

JavaScript

Java

Python

Ruby

.NET Conf China

浏览器



Playwright

Chromium

Firefox

WebKit

Selenium

Chromium

Firefox

WebKit

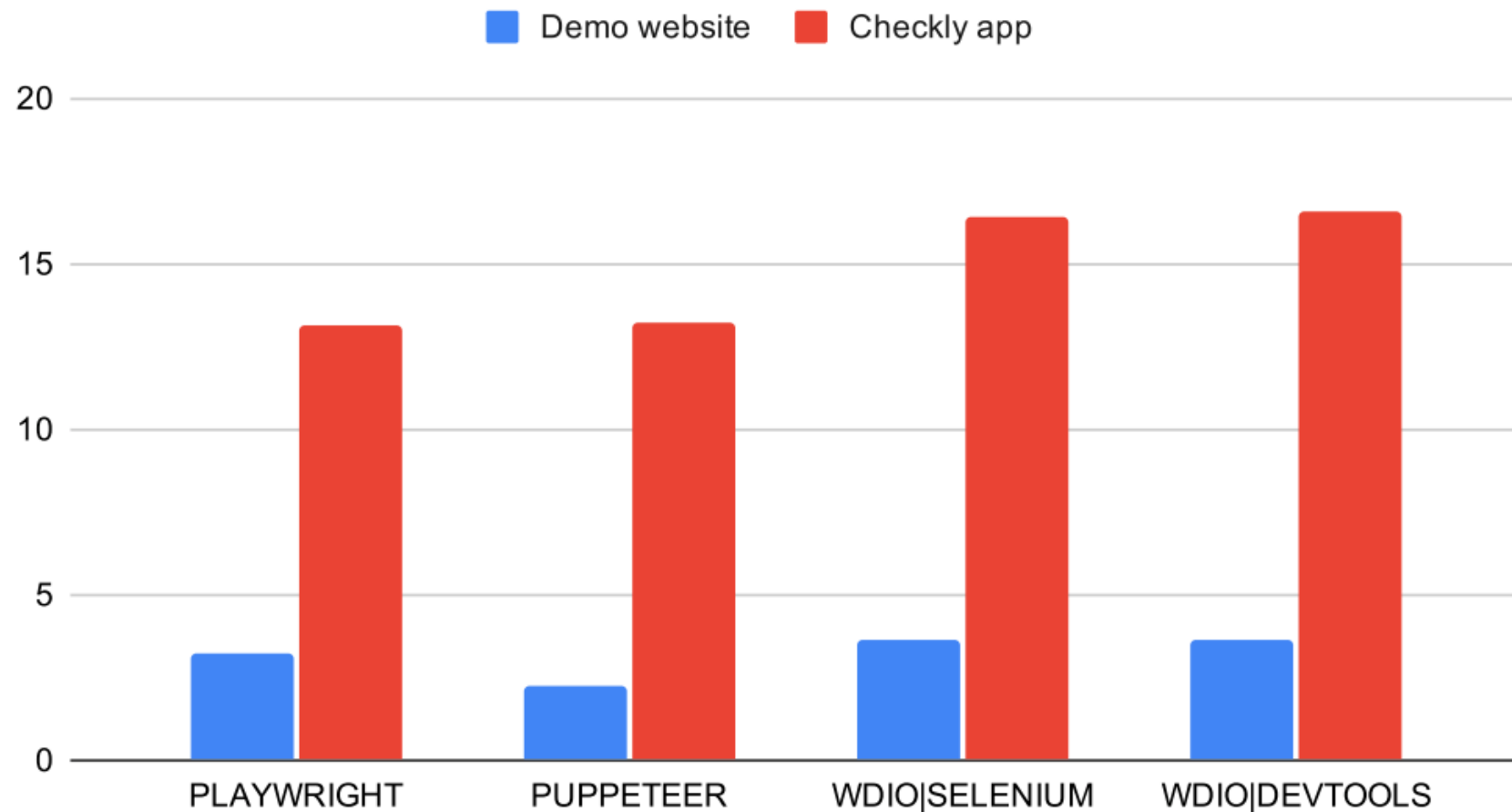
IE

.NET Conf China

性能



Mean execution timings



<https://blog.checklyhq.com/puppeteer-vs-selenium-vs-playwright-speed-comparison/#methodology-or-how->

.NET Conf China

如何使用 (NodeJS)



如何使用 - 环境准备 (NodeJS)

➤ NodeJS

➤ v14以上

➤ 代码编辑器

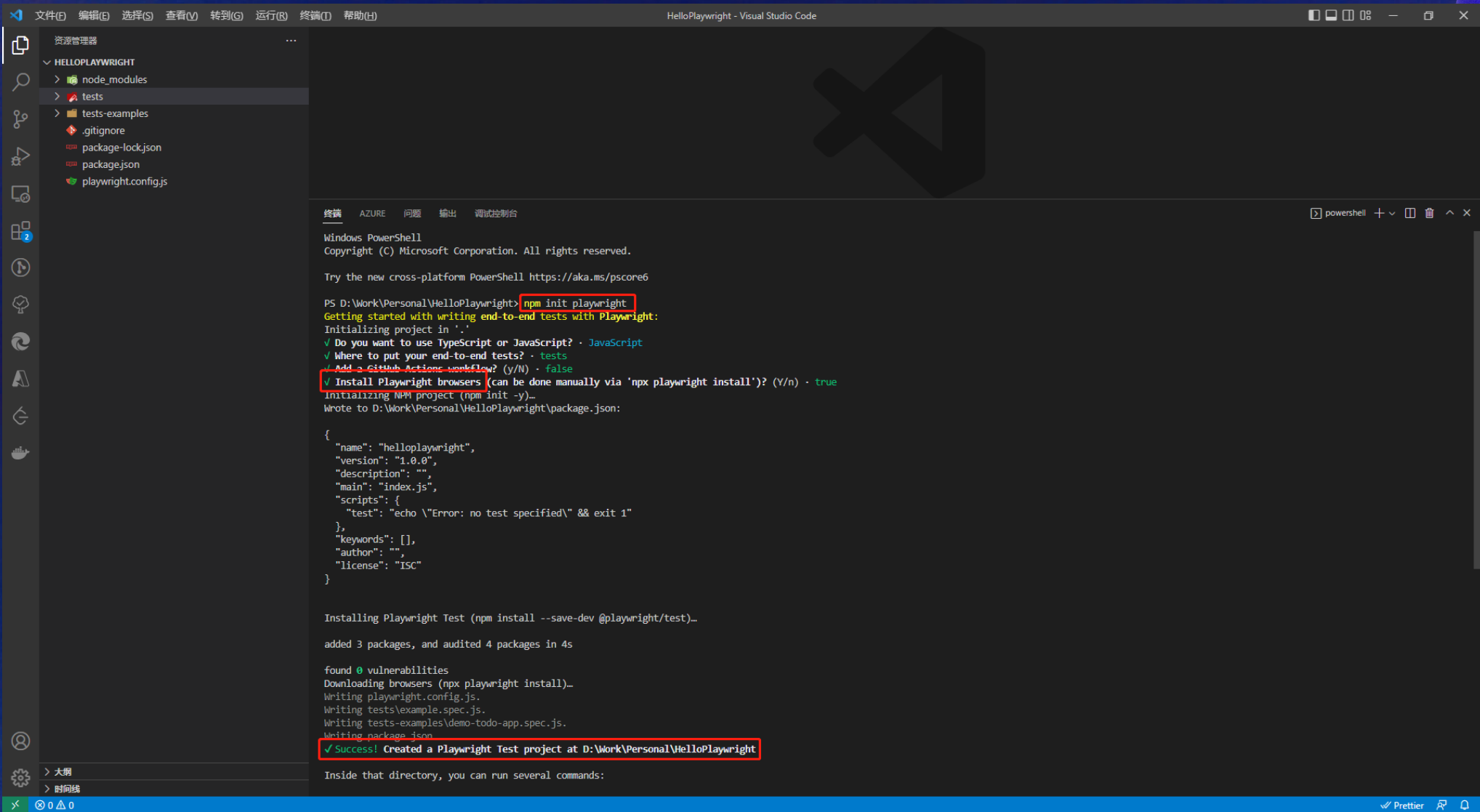
➤ VSCode

➤ Playwright

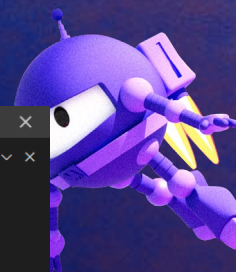
➤ `npm install playwright`



如何使用 – 项目初始化

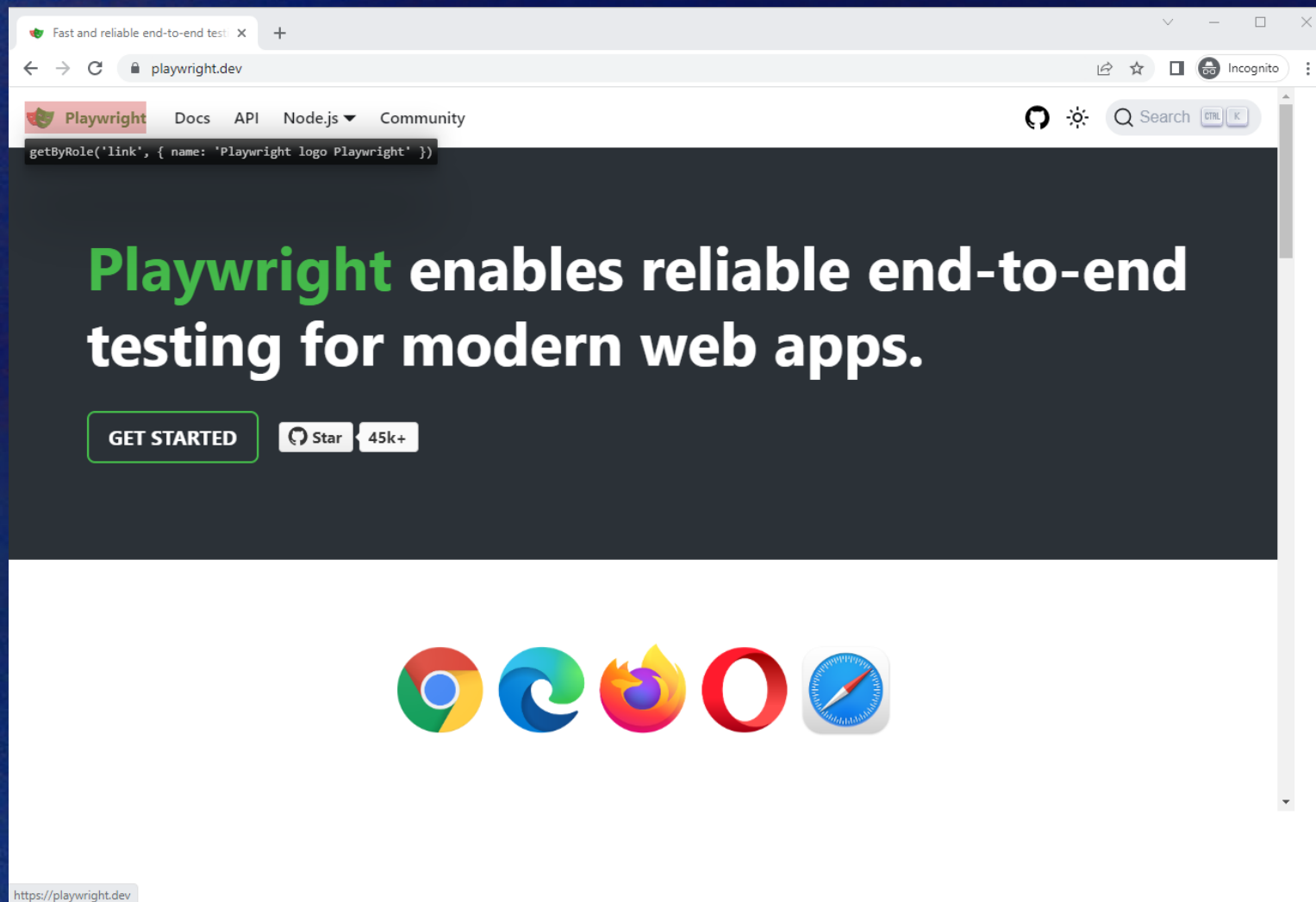
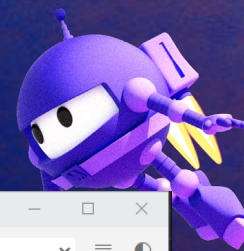


如何使用 – 录制脚本

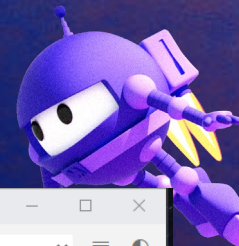


```
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H) HelloPlaywright - Visual Studio Code
资源管理器
HELLOPLAYWRIGHT
  node_modules
  tests
  tests-examples
  .gitignore
  package-lock.json
  package.json
  playwright.config.js
终端
Installing Playwright Test (npm install --save-dev @playwright/test)...
added 3 packages, and audited 4 packages in 4s
found 0 vulnerabilities
Downloading browsers (npx playwright install)...
Writing playwright.config.js.
Writing tests\example.spec.js.
Writing tests-examples\demo-todo-app.spec.js.
Writing package.json.
✓ Success! Created a Playwright Test project at D:\Work\Personal\HelloPlaywright
Inside that directory, you can run several commands:
npx playwright test
  Runs the end-to-end tests.
npx playwright test --project=chromium
  Runs the tests only on Desktop Chrome.
npx playwright test example
  Runs the tests in a specific file.
npx playwright test --debug
  Runs the tests in debug mode.
npx playwright codegen
  Auto generate tests with Codegen.
We suggest that you begin by typing:
npx playwright test
And check out the following files:
- .\tests\example.spec.js - Example end-to-end test
- .\tests-examples\demo-todo-app.spec.js - Demo Todo App end-to-end tests
- .\playwright.config.js - Playwright Test configuration
Visit https://playwright.dev/docs/intro for more information. ✨
Happy hacking! 🎉
PS D:\Work\Personal\HelloPlaywright: npx playwright codegen playwright.dev
```


如何使用 – 录制脚本



如何使用 – 录制脚本



Mouse | Playwright .NET

playwright.dev/dotnet/docs/api/class-mouse#mouse-click

Incognito

Playwright for .NET

Docs API .NET Community

API reference

Playwright

Classes

APIRequest

APIRequestContext

APIResponse

Accessibility

Browser

BrowserContext

BrowserType

ConsoleMessage

Dialog

Download

ElementHandle

FileChooser

Frame

FrameLocator

JSHandle

Mouse.ClickAsync(x, y, options)

Added in: v1.8

- `x` `<double>`
- `y` `<double>` #
- `options` `<MouseClickOptions?>`
 - `Button` `<enum MouseButton { Left, Right, Middle }?>` Defaults to `left`.
 - `ClickCount` `<int?>` defaults to 1. See `UIEvent.detail`.
 - `Delay` `<double?>` Time to wait between `mousedown` and `mouseup` in milliseconds. Defaults to 0.
- returns: `<void>`

Shortcut for `Mouse.MoveAsync(x, y, options)`, `Mouse.DownAsync(options)`, `Mouse.UpAsync(options)`.

Mouse.DblClickAsync(x, y, options)

Added in: v1.8

- `x` `<double>`
- `y` `<double>`
- `options` `<MouseDbClickOptions?>`
 - `Button` `<enum MouseButton { Left, Right, Middle }?>` Defaults to `left`.

Mouse.ClickAsync(x, y, options)

Mouse.DblClickAsync(x, y, options)

Mouse.DownAsync(options)

Mouse.MoveAsync(x, y, options)

Mouse.UpAsync(options)

Mouse.WheelAsync(deltaX, deltaY)

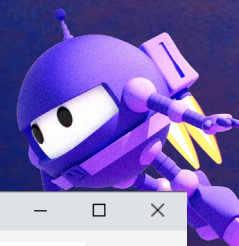
Playwright Inspector

Record

Target: Library

```
1 const { chromium } = require('playwright');
2
3 (async () => {
4   const browser = await chromium.launch({
5     headless: false
6   });
7   const context = await browser.newContext();
8   const page = await context.newPage();
9   await page.goto('https://playwright.dev/');
10  await page.getByRole('button', { name: 'Node.js' }).click();
11  await page.getByRole('navigation').getByRole('link', { name: 'Python'
12  await page.getByRole('button', { name: 'Python' }).click();
13  await page.getByRole('navigation').getByRole('link', { name: '.NET' })
14  await page.getByRole('link', { name: 'Docs' }).click();
15  await page.getByRole('navigation').filter({ hasText: 'Playwright for
16  await page.getByRole('button', { name: 'Search' }).click();
17  await page.getByPlaceholder('Search docs').click();
18  await page.getByPlaceholder('Search docs').fill('click');
19  await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, options)
20
21  // -----
22  await context.close();
23  await browser.close();
24 })();
```


如何使用 – 切换语言 (C#)

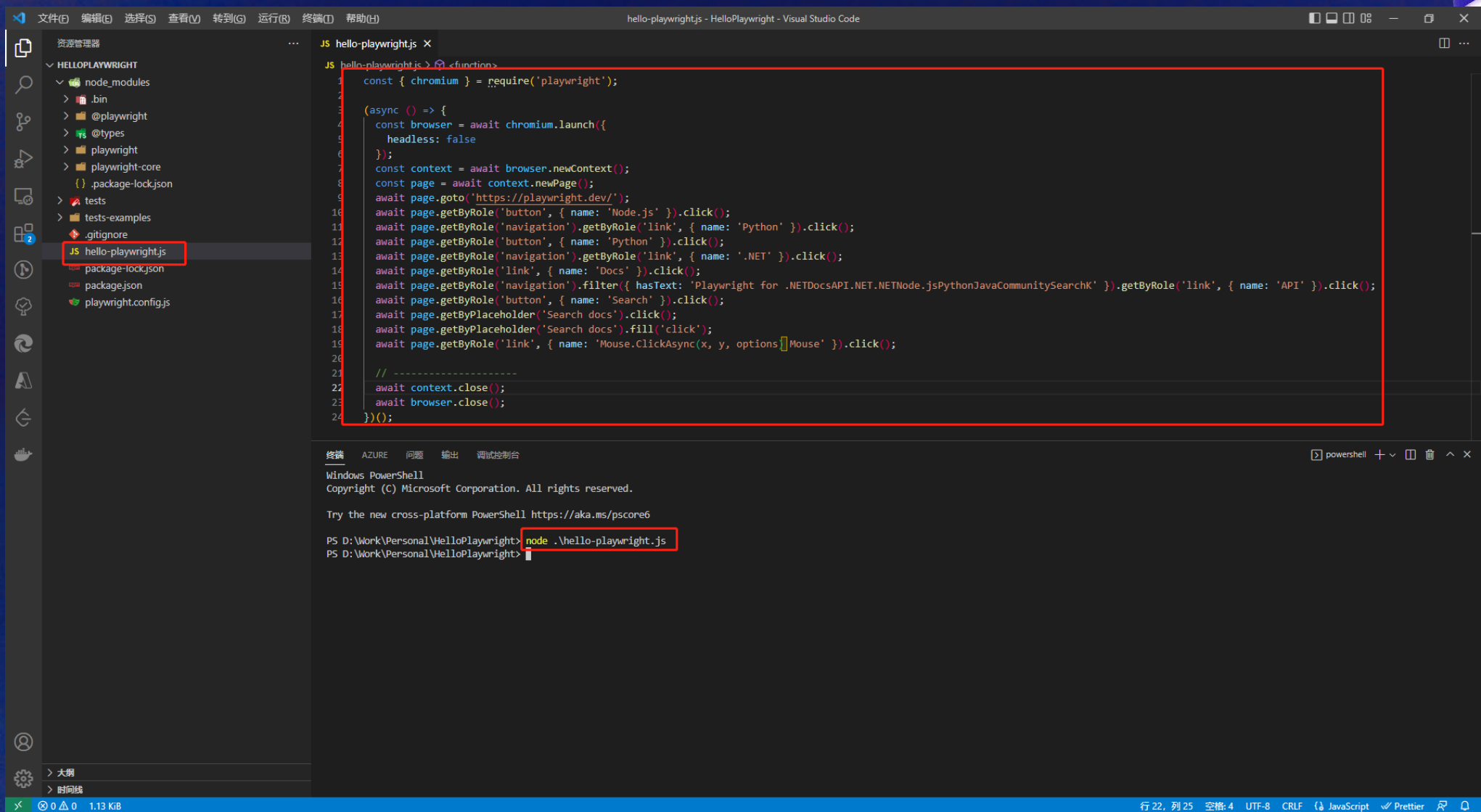


```
Playwright Inspector
Record [Icons]
1 using Microsoft.Playwright;
2 using System;
3 using System.Threading.Tasks;
4
5 class Program
6 {
7     public static async Task Main()
8     {
9         using var playwright = await Playwright.CreateAsync();
10         await using var browser = await playwright.Chromium.LaunchAsync(new BrowserTypeLaunchOptions
11         {
12             Headless = false,
13         });
14         var context = await browser.NewContextAsync();
15
16         var page = await context.NewPageAsync();
17
18         await page.GotoAsync("https://playwright.dev/");
19
20         await page.GetByRole(AriaRole.Button, new() { NameString = "Node.js" }).ClickAsync();
21
22         await page.GetByRole(AriaRole.Navigation).GetByRole(AriaRole.Link, new() { NameString = "Python" }).ClickAsync();
23
24         await page.GetByRole(AriaRole.Button, new() { NameString = "Python" }).ClickAsync();
25
26         await page.GetByRole(AriaRole.Navigation).GetByRole(AriaRole.Link, new() { NameString = ".NET" }).ClickAsync();
27
28         await page.GetByRole(AriaRole.Link, new() { NameString = "Docs" }).ClickAsync();
29
30         await page.GetByRole(AriaRole.Navigation).Filter(new() { HasTextString = "Playwright for .NETDocsAPI.NET.NETNode.jsPythonJavaCommunitySearchK" }).GetByRole(AriaRole.Link, new() { NameString = "API" }).ClickAsync();
31
32         await page.GetByRole(AriaRole.Button, new() { NameString = "Search" }).ClickAsync();
33
34         await page.GetByPlaceholder("Search docs").ClickAsync();
35
36         await page.GetByPlaceholder("Search docs").FillAsync("click");
37
38         await page.GetByRole(AriaRole.Link, new() { NameString = "Mouse.ClickAsync(x, y, options)• Mouse" }).ClickAsync();
39     }
40 }
41
42
```

Target: Library

- Node.js
- Test Runner
- Library
- Java
- Library
- Python
- Pytest
- Library
- Library Async
- .NET C#
- MSTest
- JUnit
- Library

如何使用 – 运行



如何使用 – 运行

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)

hello-playwright.js - HelloPlaywright - Visual Studio Code

资源管理器 JS hello-playwright.js X

Mouse | Playwright .NET

playwright.dev/dotnet/docs/api/class-mouse#mouse-click

Incognito

Playwright for .NET Docs API .NET Community

API reference

Playwright

Classes

APIRequest

APIRequestContext

APIResponse

Accessibility

Browser

BrowserContext

BrowserType

ConsoleMessage

Dialog

Download

ElementHandle

Mouse.ClickAsync(x, y, options)

Added in: v1.8

- `x` `<double>`
- `y` `<double>`
- `options` `<MouseClickOptions?>`
 - `Button` `<enum MouseButton { Left, Right, Middle }?>` Defaults to `left`.
 - `ClickCount` `<int?>` defaults to 1. See [UIEvent.detail](#).
 - `Delay` `<double?>` Time to wait between `mousedown` and `mouseup` in milliseconds. Defaults to 0.
- returns: `<void>`

Shortcut for `Mouse.MoveAsync(x, y, options)`, `Mouse.DownAsync(options)`, `Mouse.UpAsync(options)`.

Mouse.DblClickAsync(x, y, options)

Added in: v1.8

Mouse.ClickAsync(x, y, options)

Mouse.DblClickAsync(x, y, options)

Mouse.DownAsync(options)

Mouse.MoveAsync(x, y, options)

Mouse.UpAsync(options)

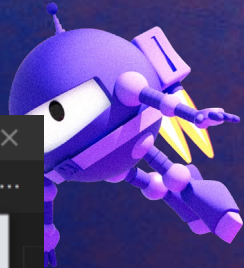
Mouse.WheelAsync(deltaX, deltaY)

大纲

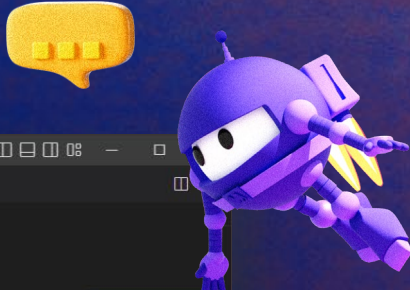
时间线

PS D:\Work\Personal\HelloPlaywright> node .\hello-playwright.js

行 9, 列 23 空格: 2 UTF-8 CRLF JavaScript Prettier



如何使用 – 截图



```
code sample.cs x
D: > LiveEvents > dotNetConf > 2022 > code sample.cs
1 const { chromium } = require('playwright');
2
3 (async () => {
4   const browser = await chromium.launch({
5     headless: true
6   });
7   const context = await browser.newContext();
8   const page = await context.newPage();
9   await page.goto('https://playwright.dev/');
10  await page.getByRole('button', { name: 'Node.js' }).click();
11  await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
12  await page.getByRole('button', { name: 'Python' }).click();
13  await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
14  await page.getByRole('link', { name: 'Docs' }).click();
15  await page.getByRole('navigation').filter({ hasText: 'Playwright for .NETDocsAPI.NET.NETNode.jsPythonJavaCommunitySearchK'
16 }).getByRole('link', { name: 'API' }).click();
17  await page.getByRole('button', { name: 'Search' }).click();
18  await page.getByPlaceholder('Search docs').click();
19  await page.getByPlaceholder('Search docs').fill('click');
20  await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, options) Mouse' }).click();
21
22  await page.screenshot({path: `Hello_Playwright.png`, fullPage: true});
23
24
25  // -----
26  await context.close();
27  await browser.close();
28 })();
29
30
```

Ln 33, Col 1 Spaces: 4 UTF-8 CRLF C# Go Live

如何使用 – 截图结果 (NodeJS)

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)

资源管理器

HELLOPLAYWRIGHT

node_modules

.bin

@playwright

@types

playwright

playwright-core

bin

lib

types

protocol.d.ts

structs.d.ts

types.d.ts

browsers.json

cli.js

index.d.ts

index.js

index.mjs

LICENSE

NOTICE

package.json

README.md

ThirdPartyNotices.txt

.package-lock.json

tests

tests-examples

.gitignore

Hello_Playwright.png

hello-playwright.js

package-lock.json

package.json

playwright.config.js

JS hello-playwright.js

1 const { chromium } = require('playwright');

2

3 (async () => {

4 const browser = await chromium.launch({

5 headless: true

6 });

7 const context = await browser.newContext();

8 const page = await context.newPage();

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

10 await page.getByRole('button', { name: 'Node.js' }).click();

11 await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();

12 await page.getByRole('button', { name: 'Python' }).click();

13 await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();

14 await page.getByRole('link', { name: 'Docs' }).click();

15 await page.getByRole('navigation').filter({ hasText: 'Playwright for .NETDocsAPI.NET' });

16 await page.getByRole('button', { name: 'Search' }).click();

17 await page.getByPlaceholder('Search docs').click();

18 await page.getByPlaceholder('Search docs').fill('click');

19 await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, options)Mouse' }).click

20

21 await page.screenshot({path: 'Hello_Playwright.png', fullPage: true});

22

23

24 // -----

25 await context.close();

26 await browser.close();

27 })();

Hello_Playwright.png

Playwright for .NET Docs API .NET Community

API reference

Playwright

Classes

APIRequest

APIRequestContext

APIResponse

Accessibility

Browser

BrowserContext

BrowserType

ConsoleMessage

Dialog

Download

ElementHandle

FileChooser

Frame

FrameLocator

JSHandle

Mouse.ClickAsync(x, y, options)

Added in v1.8

• x <double>

• y <double> #

• options <MouseClickOptions>

• Button <enum MouseButton { Left, Right, Middle }> Defaults to left.

• ClickCount <int?> defaults to 1. See UIEvent.detail.

• Delay <double?> Time to wait between mousedown and mouseup in milliseconds. Defaults to 0.

• returns <void>

Shortcut for Mouse.MoveAsync(x, y, options), Mouse.DownAsync(options), Mouse.UpAsync(options).

Mouse.DbClickAsync(x, y, options)

Added in v1.8

• x <double>

• y <double>

• options <MouseDbClickOptions>

• Button <enum MouseButton { Left, Right, Middle }> Defaults to left.

• Delay <double?> Time to wait between mousedown and mouseup in milliseconds. Defaults to 0.

• returns <void>

Shortcut for Mouse.MoveAsync(x, y, options), Mouse.DownAsync(options), Mouse.UpAsync(options), Mouse.DownAsync(options) and Mouse.UpAsync(options).

Mouse.DownAsync(options)

Added in v1.8

• options <MouseDownOptions>

• Button <enum MouseButton { Left, Right, Middle }> Defaults to left.

• ClickCount <int?> defaults to 1. See UIEvent.detail.

• returns <void>

Dispatches a mousedown event.

Mouse.MoveAsync(x, y, options)

Added in v1.8

• x <double>

• y <double>

• options <MouseMoveOptions>

• Steps <int?> Defaults to 1. Sends intermediate mousemove events.

• returns <void>

终端

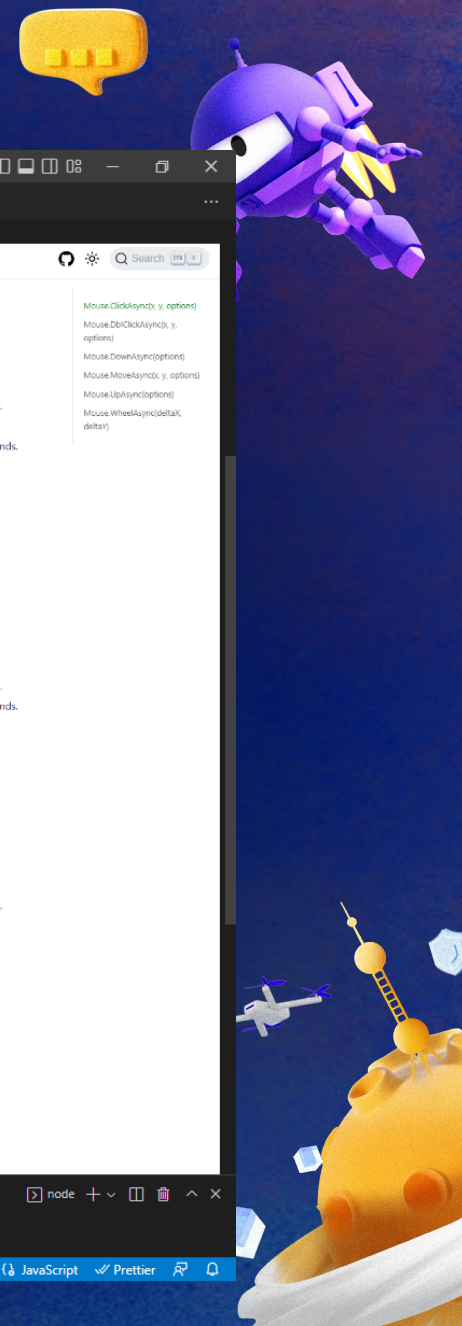
AZURE 问题 输出 调试控制台

PS D:\Work\Personal\HelloPlaywright> node .\hello-playwright.js

PS D:\Work\Personal\HelloPlaywright>

0 0 0 1.2 KiB

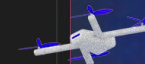
行 27, 列 6 空格 4 UTF-8 CRLF JavaScript Prettier



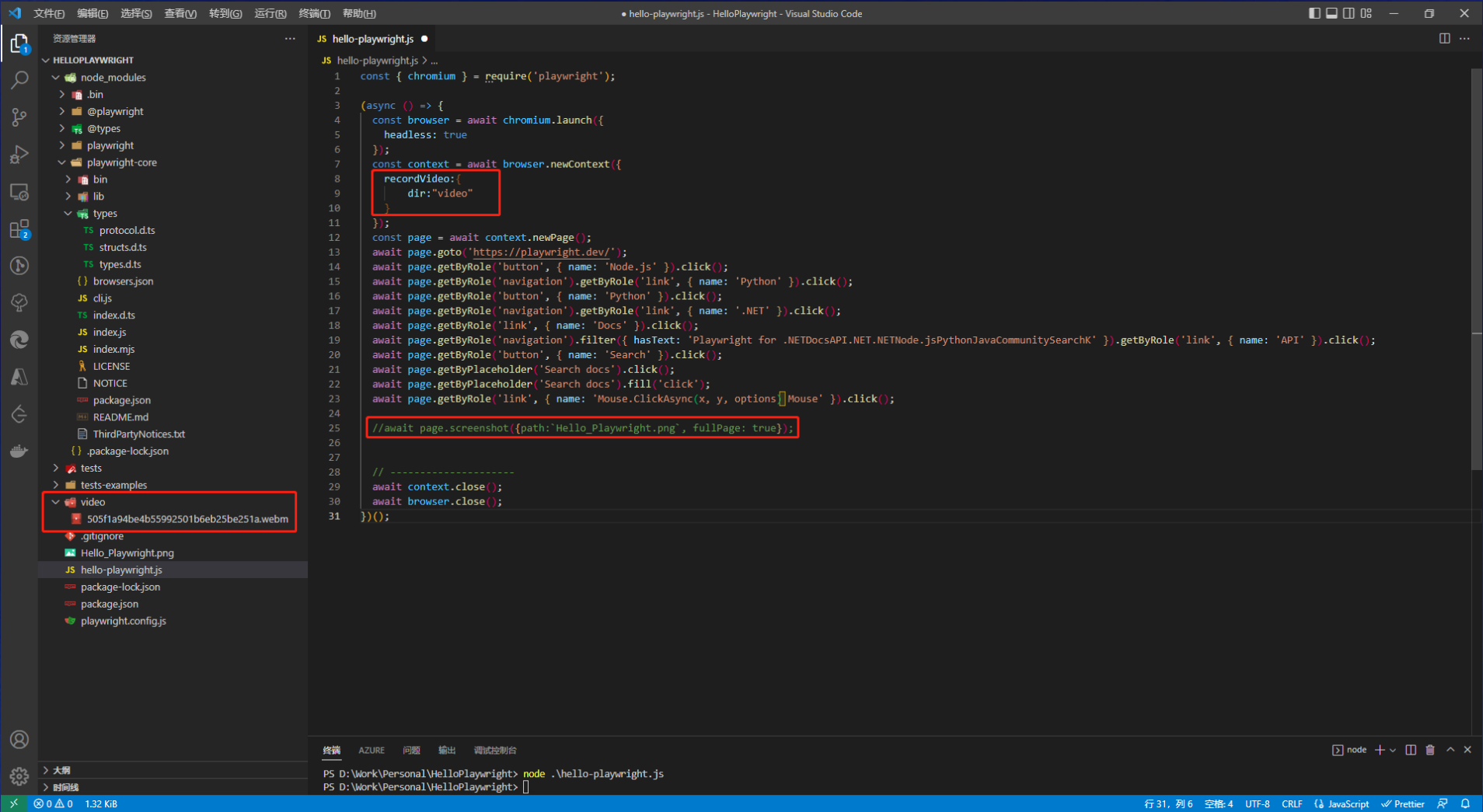
如何使用 – 视频录制



```
code sample.cs x
D:\> LiveEvents > dotNetConf > 2022 > code sample.cs
1 const { chromium } = require('playwright');
2
3 (async () => {
4   const browser = await chromium.launch({
5     headless: true
6   });
7   const context = await browser.newContext({
8     recordVideo: {
9       dir: "video"
10    }
11  });
12  const page = await context.newPage();
13  await page.goto('https://playwright.dev/');
14  await page.getByRole('button', { name: 'Node.js' }).click();
15  await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
16  await page.getByRole('button', { name: 'Python' }).click();
17  await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
18  await page.getByRole('link', { name: 'Docs' }).click();
19  await page.getByRole('navigation').filter({ hasText: 'Playwright for .NETDocsAPI.NET.NETNode.jsPythonJavaCommunitySearchK'
20  }).getByRole('link', { name: 'API' }).click();
21  await page.getByRole('button', { name: 'Search' }).click();
22  await page.getByPlaceholder('Search docs').click();
23  await page.getByPlaceholder('Search docs').fill('click');
24  await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, options) Mouse' }).click();
25
26  //await page.screenshot({path: `Hello_Playwright.png`, fullPage: true});
27
28  // -----
29  await context.close();
30  await browser.close();
31 })();
```



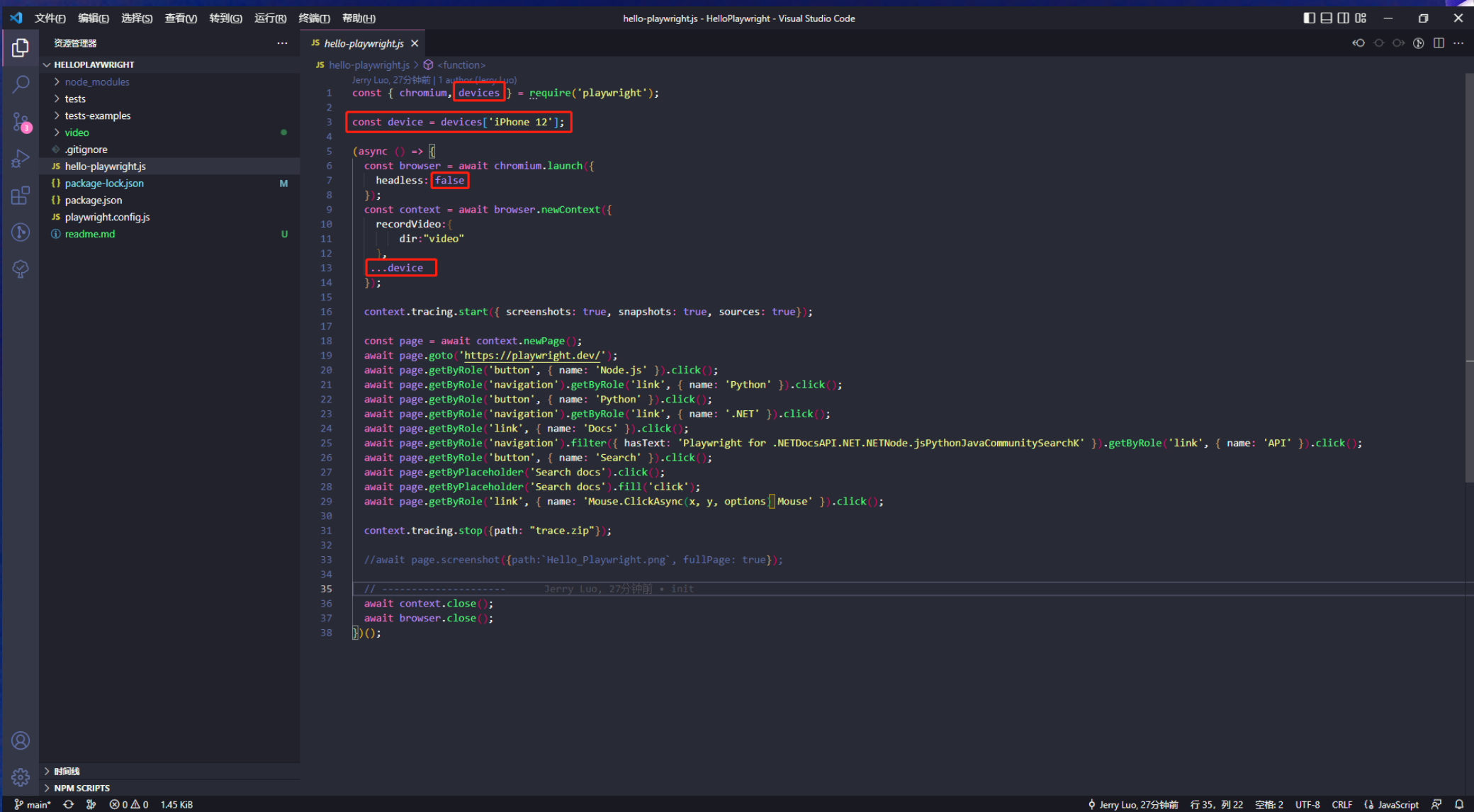
如何使用 – 视频录制运行结果



如何使用 – 视频



如何使用 – 设备模拟



如何使用 – 设备模拟

资源管理器

HELLOPLAYWRIGHT

node_modules

tests

tests-examples

video

.gitignore

JS hello-playwright.js

package-lock.json

package.json

playwright.config.js

readme.md

JS hello-playwright.js

```
JS hello-playwright.js <function>
Jerry Luo, 29分钟前 | 1 author (Jerry Luo)
1  const { chromium, devices } = require('playwright');
2
3  const device = devices['iPhone 12'];
4
5  (async () => {
6    const browser = await chromium.launch({
7      headless: false
8    });
9    const context = await browser.newContext({
10     recordVideo: {
11       dir: "video"
12     },
13     ...device
14   });
15
16   context.tracing.start({ screenshots: true, snapshots: true, sources: true });
17
18   const page = await context.newPage();
19   await page.goto('https://playwright.dev/');
20   await page.getByRole('button', { name: 'Node.js' }).click();
21   await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
22   await page.getByRole('button', { name: 'Python' }).click();
23   await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
24   await page.getByRole('link', { name: 'Docs' }).click();
25   await page.getByRole('navigation').filter({ hasText: 'Playwright for .NETDocsAPI.NET.'
26   await page.getByRole('button', { name: 'Search' }).click();
27
28   })
29 }
```

问题

终端

调试控制台

输出

GIT LENS

Windows PowerShell

版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell <https://aka.ms/pscore6>

PS D:\Work\Personal\NETConf2022-Playwright\HelloPlaywright> node .\hello-playwright.js

node:internal/process/promises:288

triggerUncaughtException(err, true /* fromPromise */);

^

locator.click: Timeout 30000ms exceeded.

===== logs =====

waiting for getByRole('button', { name: 'Node.js' })

=====

at D:\Work\Personal\NETConf2022-Playwright\HelloPlaywright\hello-playwright.js:20:55 {

name: 'TimeoutError'

}

Node.js v18.12.1

PS D:\Work\Personal\NETConf2022-Playwright\HelloPlaywright> node .\hello-playwright.js

main

0 0 0

1.45 KiB

Fast and reliable end-to-end

playwright.dev

无痕模式

Playwright

Playwright enables reliable end-to-end testing for modern web apps.

GET STARTED

Star 45k+

Google

Microsoft

Apple

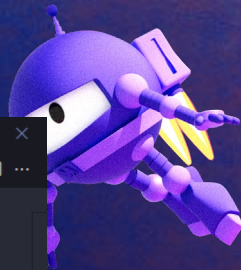
Firefox

Opera

Brave

powerShell

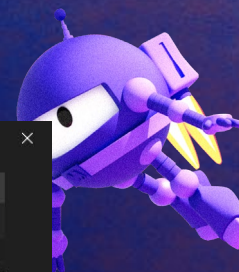
'API' }).click();



常见问题



常见问题 – Cannot find module 'playwright'



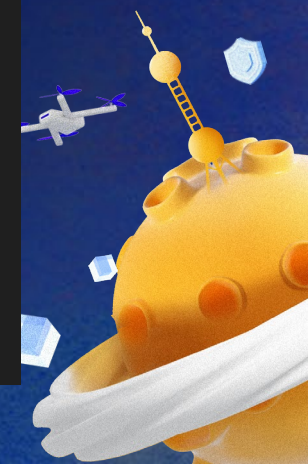
The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders like 'node_modules', 'tests', and 'tests-examples'. The main editor area displays a JavaScript file 'hello-playwright.js' with a single line of code: `const { chromium } = require('playwright');`. Below the editor, the Output window shows a PowerShell terminal session. The terminal output includes a 'Happy hacking!' message, a command to run the script, and a detailed error message: 'Error: Cannot find module 'playwright''. The error stack trace shows the module resolution process. At the bottom of the terminal, the command `npm install playwright` is entered.

```
JS hello-playwright.js X
JS hello-playwright.js > ...
1 const { chromium } = require('playwright');
```

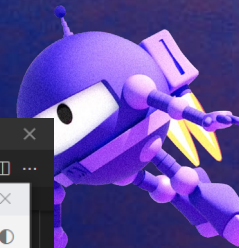
Happy hacking! 🎉

```
PS D:\Work\Personal\HelloPlaywright> node .\hello-playwright.js
node:internal/modules/cjs/loader:959
  throw err;
  ^

Error: Cannot find module 'playwright'
Require stack:
- D:\Work\Personal\HelloPlaywright\hello-playwright.js
    at Function.Module._resolveFilename (node:internal/modules/cjs/loader:956:15)
    at Function.Module._load (node:internal/modules/cjs/loader:884:27)
    at Module.require (node:internal/modules/cjs/loader:1028:19)
    at require (node:internal/modules/cjs/helpers:102:18)
    at Object.<anonymous> (D:\Work\Personal\HelloPlaywright\hello-playwright.js:1:22)
    at Module._compile (node:internal/modules/cjs/loader:1126:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1180:10)
    at Module.load (node:internal/modules/cjs/loader:1004:32)
    at Function.Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [ 'D:\Work\Personal\HelloPlaywright\hello-playwright.js' ]
}
PS D:\Work\Personal\HelloPlaywright> npm install playwright
```



常见问题 – 如何使用Playwright Inspector



The screenshot displays the Playwright Inspector interface within Visual Studio Code. The main window shows the Playwright website with the text "Playwright enables reliable end-to-end testing for modern web apps." and a "GET STARTED" button. The Playwright Inspector panel on the right shows a test script with the following code:

```
1 const { chromium } = require('playwright');
2
3 (async () => {
4   const browser = await chromium.launch({
5     headless: true
6   });
7   const context = await browser.newContext({
8     recordVideo: {
9       dir: "video"
10    }
11  });
12  const page = await context.newPage();
13  await page.goto('https://playwright.dev/');
14  await page.getByRole('button', { name: 'Node.js' }).click();
15  await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
16  await page.getByRole('button', { name: 'Python' }).click();
17  await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
18  await page.getByRole('link', { name: 'Docs' }).click();
19  await page.getByRole('navigation').filter({ hasText: 'Playwright for' }).getByRole('button', { name: 'Search' }).click();
20  await page.getByRole('button', { name: 'Search' }).click();
21  await page.getByPlaceholder('Search docs').click();
22  await page.getByPlaceholder('Search docs').fill('click');
23  await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, option' }).click();
24
25  await page.locator('li:nth-child(3) > code').first().click();
26
```

The Playwright Inspector panel also shows the following log:

- > browserContext.newPage ✓ — 946ms
- > page.goto(https://playwright.dev/) ✓ — 647ms
- > page.getByRole('button', { name: 'Node.js' }).click() II
 - waiting for getByRole('button', { name: 'Node.js' })
 - locator resolved to visible Node.js
 - attempting click action
 - waiting for element to be visible, enabled and stable
 - element is visible, enabled and stable
 - scrolling into view if needed
 - done scrolling

The bottom status bar shows the file path: `PS D:\Work\Personal\HelloPlaywright> $env:NODEBUG=1` and the command: `node .\hello-playwright.js`.

常见问题 – 如何使用Playwright Inspector

The screenshot displays the Playwright Inspector interface within Visual Studio Code. The main window shows a web browser with the Playwright website. The 'Elements' panel on the right highlights a link element with the role 'link'. The 'Console' panel shows the test script execution. The 'Playwright Inspector' panel on the right shows the test script and the current step's details.

Playwright Inspector Console:

```
const { chromium } = require('playwright');
(async () => {
  const browser = await chromium.launch({
    headless: true
  });
  const context = await browser.newContext({
    recordVideo: {
      dir: "video"
    }
  });
  const page = await context.newPage();
  await page.goto('https://playwright.dev/');
  await page.getByRole('button', { name: 'Node.js' }).click();
  await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
  await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
  await page.getByRole('link', { name: 'Docs' }).click();
  await page.getByRole('navigation').filter({ hasText: 'Playwright for' }).click();
  await page.getByRole('button', { name: 'Search' }).click();
  await page.getByPlaceholder('Search docs').fill('click');
  await page.getByPlaceholder('Search docs').fill('click');
  await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, option' }).click();
  await page.locator('li:nth-child(3) > code').first().click();
})
```

Playwright Inspector Details:

Target: hello-playwright.js

getByRole('button', { name: 'Node.js' })

browserContext.newPage ✓ — 946ms

page.goto(https://playwright.dev/) ✓ — 647ms

page.getByRole('button', { name: 'Node.js' }).click() II

waiting for getByRole('button', { name: 'Node.js' })

locator resolved to visible Node.js

attempting click action

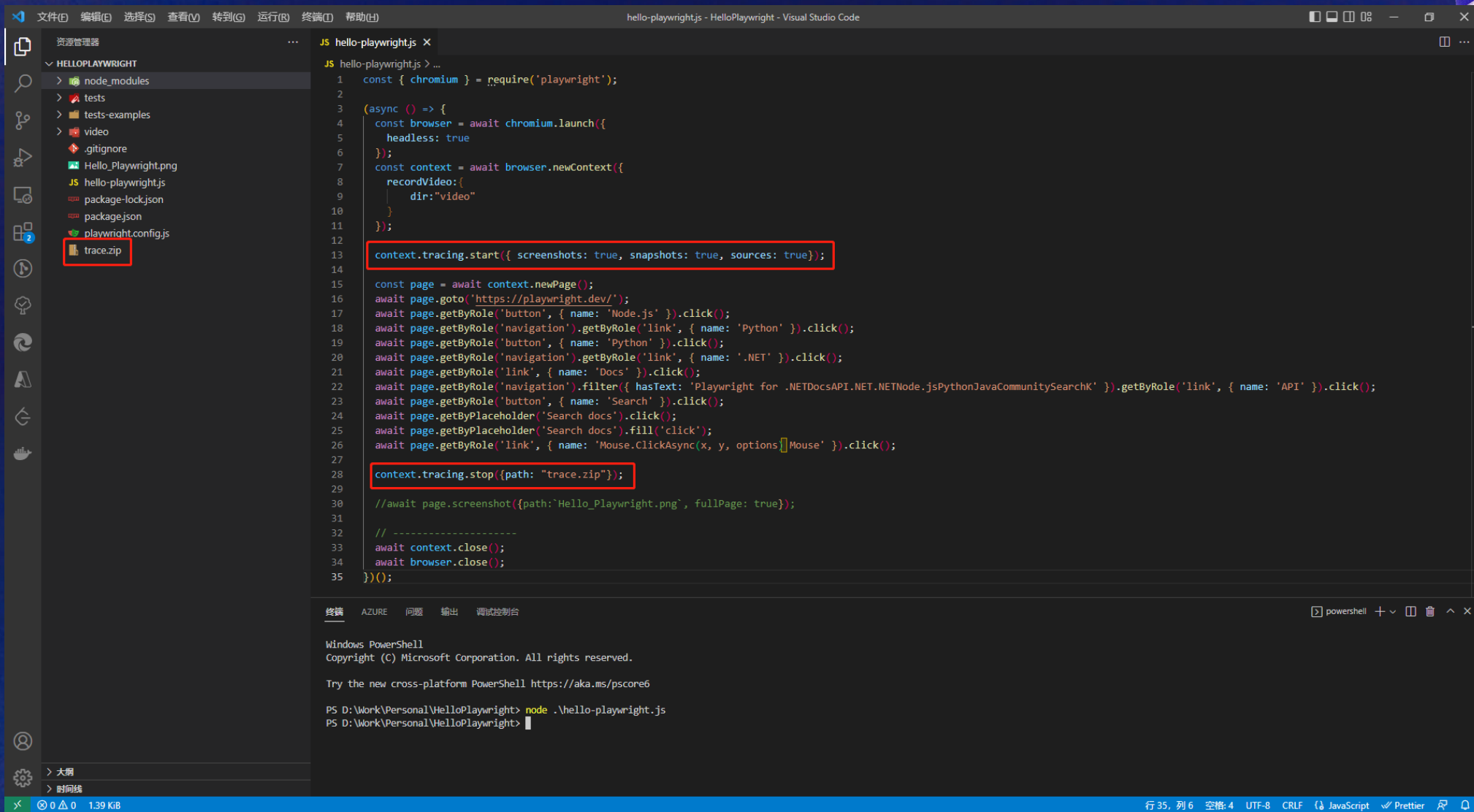
waiting for element to be visible, enabled and stable

element is visible, enabled and stable

scrolling into view if needed

done scrolling

常见问题 – 如何使用Trace Viewer



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'HELLOPLAYWRIGHT' with various files and folders. The code editor displays a JavaScript file named 'hello-playwright.js' with the following code:

```
1  const { chromium } = require('playwright');
2
3  (async () => {
4    const browser = await chromium.launch({
5      headless: true
6    });
7    const context = await browser.newContext({
8      recordVideo: {
9        dir: "video"
10     };
11   });
12
13   context.tracing.start({ screenshots: true, snapshots: true, sources: true });
14
15   const page = await context.newPage();
16   await page.goto('https://playwright.dev/');
17   await page.getByRole('button', { name: 'Node.js' }).click();
18   await page.getByRole('navigation').getByRole('link', { name: 'Python' }).click();
19   await page.getByRole('button', { name: 'Python' }).click();
20   await page.getByRole('navigation').getByRole('link', { name: '.NET' }).click();
21   await page.getByRole('link', { name: 'Docs' }).click();
22   await page.getByRole('navigation').filter({ hasText: 'Playwright for .NETDocsAPI.NET.NETNode.jsPythonJavaCommunitySearchK' }).getByRole('link', { name: 'API' }).click();
23   await page.getByRole('button', { name: 'Search' }).click();
24   await page.getByPlaceholder('Search docs').click();
25   await page.getByPlaceholder('Search docs').fill('click');
26   await page.getByRole('link', { name: 'Mouse.ClickAsync(x, y, options | Mouse' }).click();
27
28   context.tracing.stop({ path: "trace.zip" });
29
30   //await page.screenshot({path: 'Hello_Playwright.png', fullPage: true});
31
32   // -----
33   await context.close();
34   await browser.close();
35 })();
```

The file explorer on the left shows the following files and folders:

- node_modules
- tests
- tests-examples
- video
- .gitignore
- Hello_Playwright.png
- hello-playwright.js
- package-lock.json
- package.json
- playwright.config.js
- trace.zip

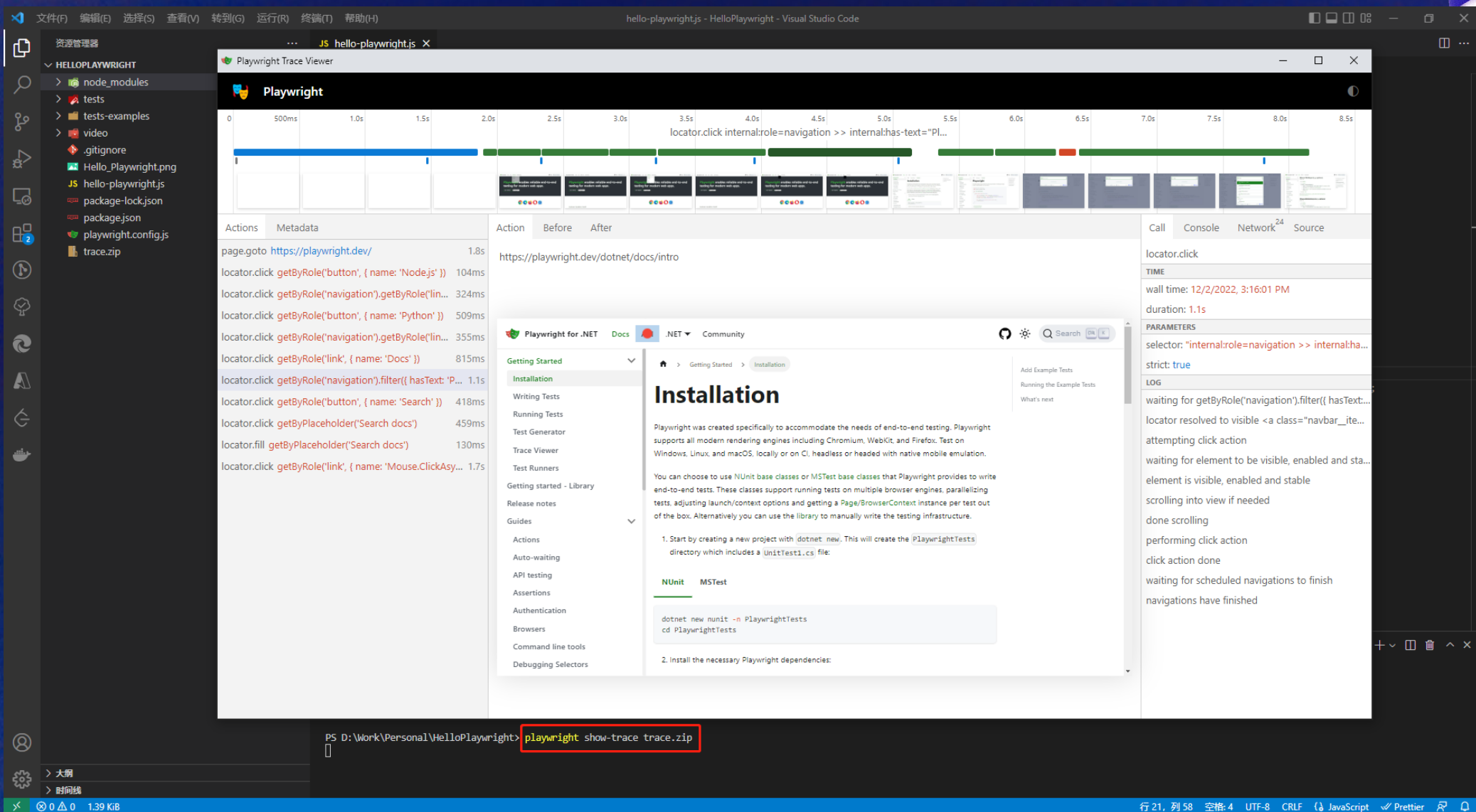
The terminal at the bottom shows the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Work\Personal\HelloPlaywright> node .\hello-playwright.js
PS D:\Work\Personal\HelloPlaywright>
```

常见问题 – 如何使用Trace Viewer



The screenshot displays the Visual Studio Code interface with the Playwright Trace Viewer open. The trace viewer shows a timeline of actions performed during a test run, including page.goto, locator.click, and locator.fill. The terminal at the bottom shows the command `playwright show-trace trace.zip` being executed.

Playwright Trace Viewer

Actions Metadata

Action	Before	After
page.goto https://playwright.dev/		1.8s
locator.click getByRole('button', { name: 'Node.js' })		104ms
locator.click getByRole('navigation').getByRole('link', { name: 'Python' })		324ms
locator.click getByRole('button', { name: 'Python' })		509ms
locator.click getByRole('navigation').getByRole('link', { name: 'Docs' })		355ms
locator.click getByRole('link', { name: 'Docs' })		815ms
locator.click getByRole('navigation').filter({ hasText: 'P...' })		1.1s
locator.click getByRole('button', { name: 'Search' })		418ms
locator.click getByPlaceholder('Search docs')		459ms
locator.fill getByPlaceholder('Search docs')		130ms
locator.click getByRole('link', { name: 'Mouse.ClickAsy...' })		1.7s

Installation

Playwright was created specifically to accommodate the needs of end-to-end testing. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox. Test on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation.

You can choose to use **JUnit** base classes or **MSTest** base classes that Playwright provides to write end-to-end tests. These classes support running tests on multiple browser engines, parallelizing tests, adjusting launch/context options and getting a `Page/BrowserContext` instance per test out of the box. Alternatively you can use the library to manually write the testing infrastructure.

- Start by creating a new project with `dotnet new`. This will create the `PlaywrightTests` directory which includes a `UnitTest1.cs` file:

```
dotnet new junit -n PlaywrightTests
cd PlaywrightTests
```

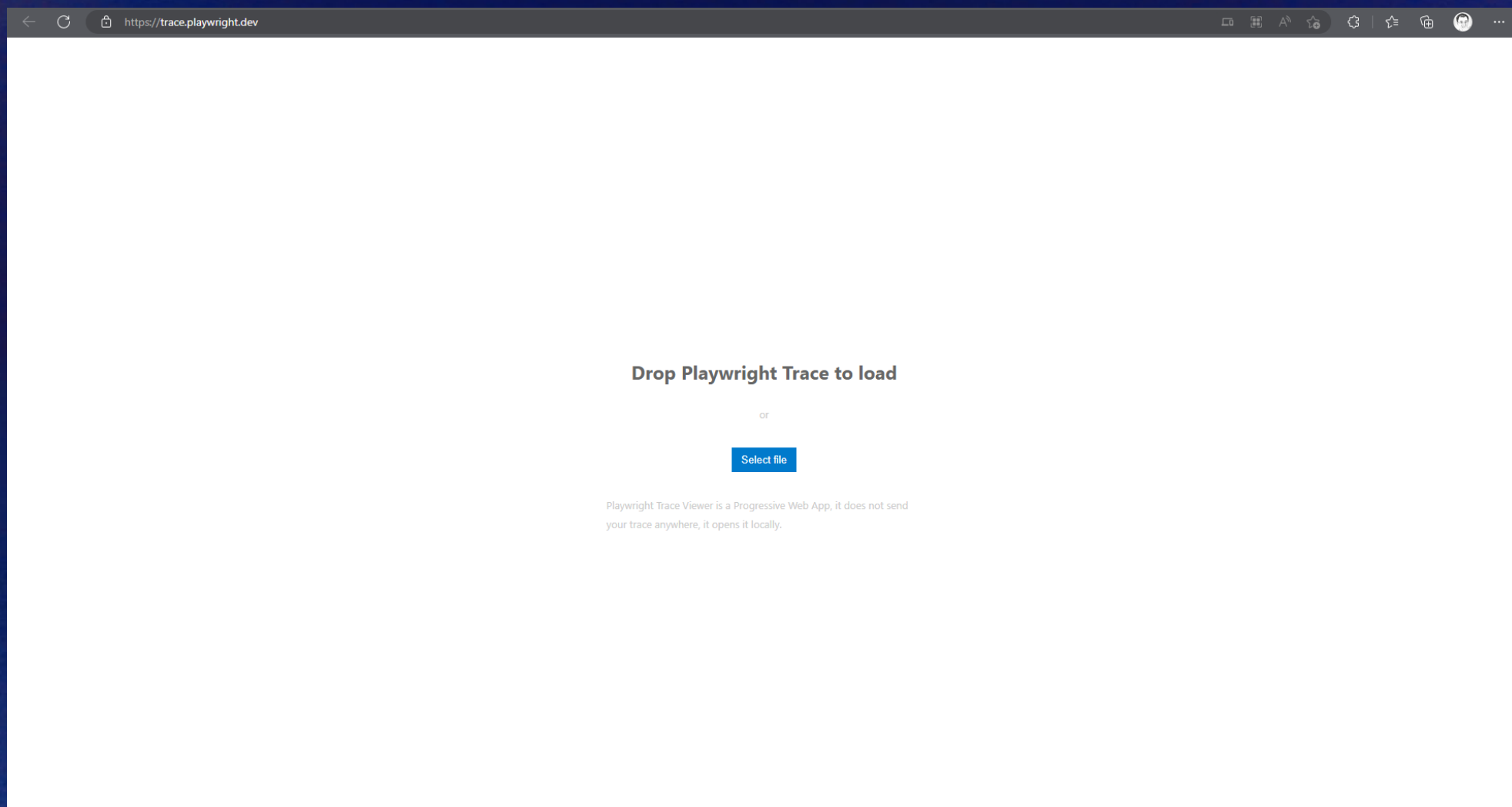
- Install the necessary Playwright dependencies:

Terminal

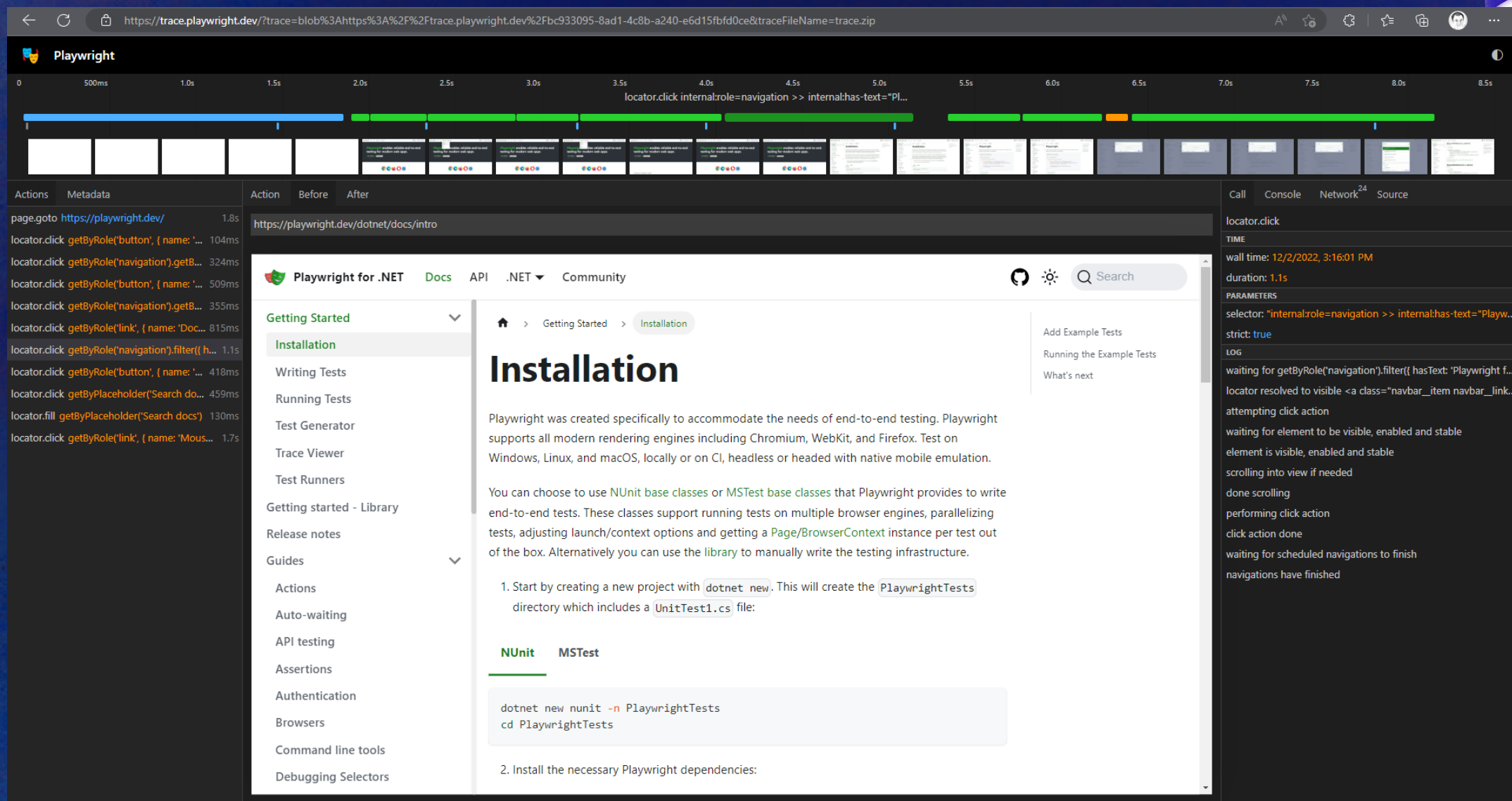
```
PS D:\Work\Personal\HelloPlaywright> playwright show-trace trace.zip
```


常见问题 – 如何使用Trace Viewer

<https://trace.playwright.dev>



常见问题 – 如何使用Trace Viewer



The screenshot displays the Playwright Trace Viewer interface, which is used to inspect the execution of a test run. The interface is divided into several sections:

- Top Bar:** Shows the Playwright logo and the URL of the trace file: `https://trace.playwright.dev/?trace=blob%3Ahttps%3A%2F%2Ftrace.playwright.dev%2Fbc933095-8ad1-4c8b-a240-e6d15bfd0ce&traceFileName=trace.zip`.
- Timeline:** A horizontal bar at the top of the main area showing the duration of the test run, with a time scale from 0 to 8.5s. A specific action, `locator.click internalrole=navigation >> internalhas-text="PL...`, is highlighted.
- Actions Panel:** Located on the left, it lists the actions performed during the test run, such as `page.goto https://playwright.dev/` and `locator.click getByRole('button', { name: '...'}`, along with their durations.
- Trace Viewer:** The central area displays a screenshot of the web page being tested, which is the Playwright for .NET documentation page. The URL `https://playwright.dev/dotnet/docs/intro` is visible in the address bar.
- Call Console:** On the right, it shows the sequence of events during the test run, including `locator.click`, `TIME`, `wall time: 12/2/2022, 3:16:01 PM`, `duration: 1.1s`, and `PARAMETERS`.

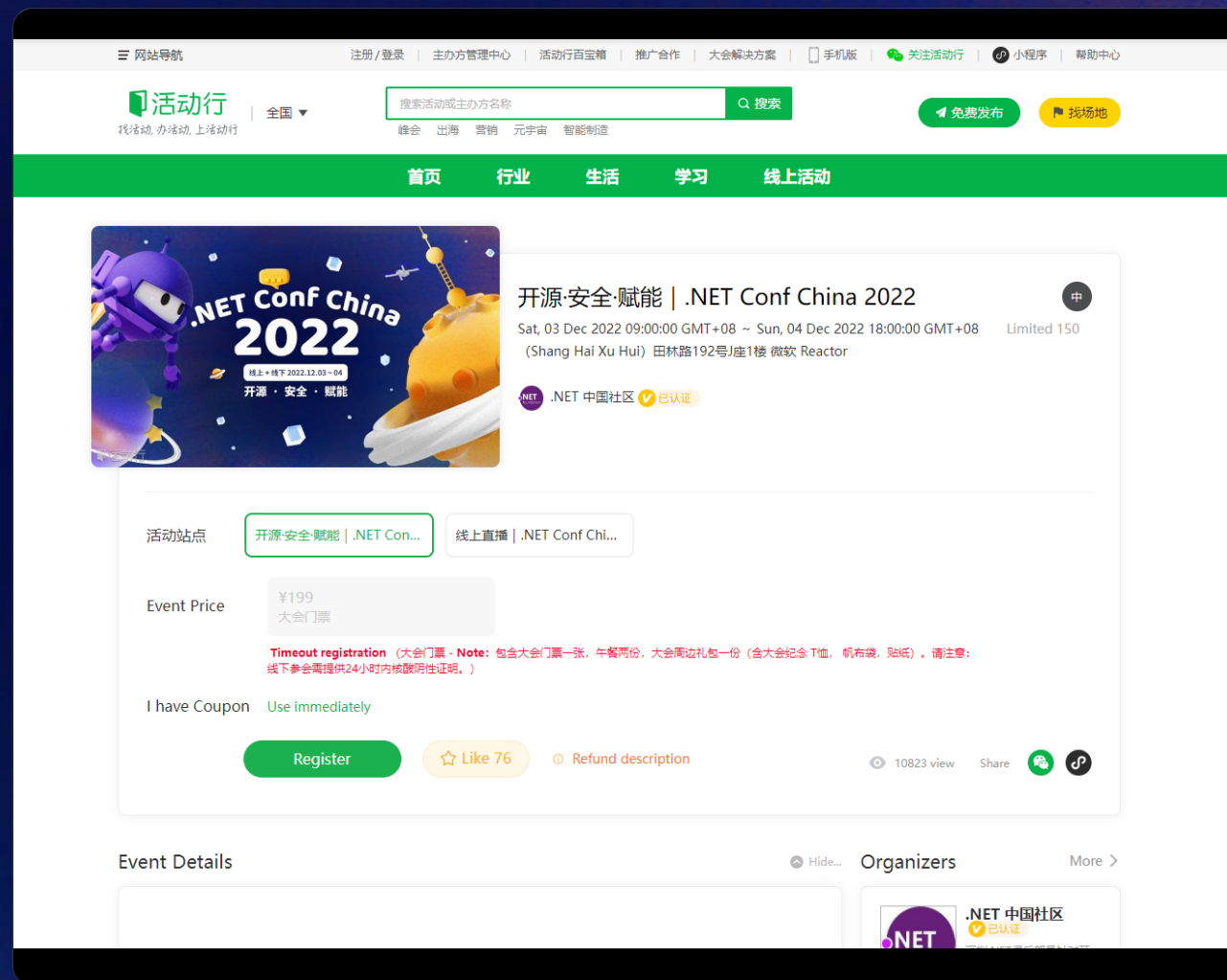
Talk is cheap, show me the code



Demo – 单元测试 (C#)

NET Conf China 2022 活动页微信扫码登录

1. 点击登录按钮
2. 选择微信扫码登录
3. 二维码截图
4. 本地打开截图
5. 微信扫码
6. 跳回当前页, 根据用户信息验证是否成功



Demo - 环境准备 (C#)

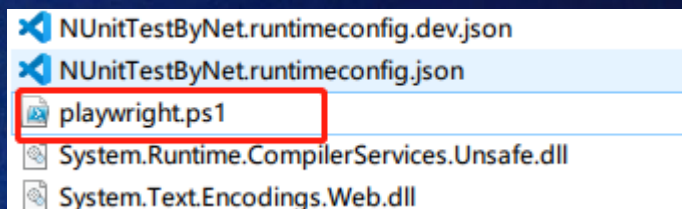
➤ 创建单元测试项目

- NUnit
- MSTest

➤ 安装Playwright相关依赖

- Microsoft.Playwright.NUnit
- Microsoft.Playwright.MSTest

➤ (Optional) 安装浏览器



Demo - 环境准备 (C#)

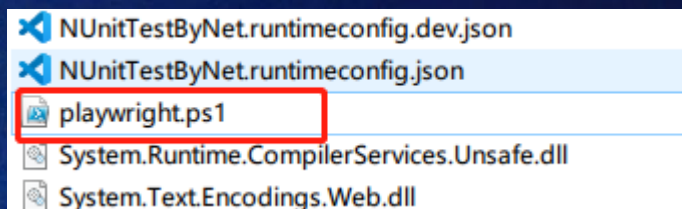
➤ 创建单元测试项目

- NUnit
- MSTest

➤ 安装Playwright相关依赖

- Microsoft.Playwright.NUnit
- Microsoft.Playwright.MSTest

➤ (Optional) 安装浏览器



Demo – 代码

0 个引用 | 0 项更改 | 0 名作者, 0 项更改

```
public class Tests
{
    [Parallelizable(ParallelScope.Self)]
    [TestFixture]
    0 个引用 | 0 项更改 | 0 名作者, 0 项更改
    public class Tests01 : PageTest
    {
        [Test]
        ✓ | 0 个引用 | 0 项更改 | 0 名作者, 0 项更改
        public async Task TestLoginInNETConfChina2022Page()
        {
            await Page.GotoAsync("https://www.huodongxing.com/event/4674917451700");
            //点击登录按钮
            await Page.GetByRole(AriaRole.Link, new() { NameString = "登录" })
                .Filter(new() { HasTextString = "登录" }).ClickAsync();

            //点击微信登录按钮, 并且等待, 直到二维码获取完成
            await Page.RunAndWaitForRequestAsync(async () =>
            {
                await Page.GetByRole(AriaRole.Link, new() { NameString = "微信" }).ClickAsync();
            }, new Regex(".*qrcode.*"));

            //页面截图
            string screenshotPath = $"LoginQRCode{DateTime.Now.Ticks}.png";
            await Page.ScreenshotAsync(new PageScreenshotOptions() { Path = screenshotPath, FullPage = true });

            //打开截图等待扫码
            string qrCodeFilePath = $"{Directory.GetCurrentDirectory()}\\{screenshotPath}";
            OpenQRCodeAsync(qrCodeFilePath);

            //等待扫码成功后, 回到NET Conf 2022 活动页面
            await Page.WaitForURLAsync("https://www.huodongxing.com/event/4674917451700",
                new PageWaitForURLOptions() { Timeout = 0 });

            //通过用户信息来诊断是否登录成功
            var currentUserInfo = Page.GetByRole(AriaRole.Link, new() { NameString = "Hi 阿白" });
            await Expect(currentUserInfo).ToBeVisibleAsync();
        }
    }
}
```

0 个引用 | 0 项更改 | 0 名作者, 0 项更改

```
public override BrowserNewContextOptions ContextOptions()
{
    //设置视频保存目录
    return new BrowserNewContextOptions() { RecordVideoDir = "video" };
}
```

```
private void OpenQRCodeAsync(string qrCodeFilePath)
{
    //建立新的系统进程
    System.Diagnostics.Process process = new System.Diagnostics.Process();
    //设置文件名, 此处为图片的真实路径+文件名
    process.StartInfo.FileName = qrCodeFilePath;
    //此为关键部分。设置进程运行参数, 此时为最大化窗口显示图片。
    process.StartInfo.Arguments = "rundll32.exe %systemroot%\\system32\\shimgvw.dll,ImageView_Fullscreen";
    //此项为是否使用Shell执行程序, 因系统默认为true, 此项也可不设, 但若设置必须为true
    process.StartInfo.UseShellExecute = true;
    //此处可以更改进程所打开窗体的显示样式, 可以不设
    process.StartInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
    process.Start();
    process.Close();
}
```

Demo – 运行



```
//打开截图等待扫码
string qrCodeFilePath = $"{ Directory.GetCurrentDirectory()}\{screenShotPath}";
OpenQRCodeAsync(qrCodeFilePath);

//等待扫码成功后,回到NET Conf 2022 活动页面
await Page.WaitForURLAsync($"https://www.huodongxing.com/event/4674917451700",
    new PageWaitForURLOptions() { Timeout = 0 });
```

LoginQRCode63805685239...

使用微信扫一扫登录

「活动行」

测试资源管理器

1 0

1 0

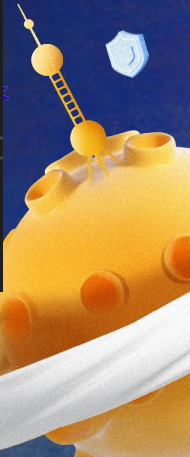
测试	持续时间	特征
✔ NUnitByNet (1)	18.9 秒	
✔ NUnitByNet (1)	18.9 秒	
✔ Tests+Tests01 (1)	18.9 秒	
✔ TestLoginInNETConfChina20...	18.9 秒	

测试详细信息摘要

✔ TestLoginInNETConfChina2022Page

源: [UnitTest1.cs](#) 行 18

⌚ 持续时间: 18.9 秒



Demo – 视频





B站



抖音



公众号



微信群

Thank you!

Let's build amazing apps with .NET 7
get.dot.net/7

