Engineer

Playwright

інструмент, що керує браузером у тестах

Що це таке

Playwright — фреймворк з відкритим кодом від Microsoft для автоматизації браузерів з єдиним програмним інтерфейсом для Chromium, Firefox і WebKit. Playwright Test, опублікований у npm як `@playwright/test`, — це тестовий раннер, побудований на ньому, і саме його найчастіше мають на увазі, коли вживають цю назву. Той самий проєкт випускає звичайну бібліотеку для скриптів, які не є тестами, інструмент командного рядка й MCP-сервер для ШІ-агентів, а також розширення для Visual Studio Code, усе за ліцензією Apache License 2.0.

Окрім JavaScript і TypeScript, те саме керування браузером доступне з Python, де проєкт радить свій плагін для pytest, з Java разом із будь-яким тестовим фреймворком, як-от JUnit, і з .NET з базовими класами для MSTest, NUnit і xUnit.

`npx playwright install` завантажує власні збірки трьох рушіїв, і оскільки кожен реліз оновлює ці версії, після оновлення команду може знадобитися запустити знову. Playwright також може керувати Google Chrome і Microsoft Edge, якщо вони вже встановлені на машині. За замовчуванням тести виконуються в режимі headless, без видимого вікна. Приклад у посібнику зі встановлення запускає тести паралельно в усіх трьох рушіях через налаштовані браузерні проєкти; без цих проєктів типовим браузером засобу запуску тестів є Chromium. Кожен тест отримує свіжий контекст браузера, рівнозначний новому профілю, тож cookie та сховище одного тесту не переходять до наступного.

Тест читається як кроки, які зробила б людина:

```
import { test, expect } from "@playwright/test";

test("після входу відкриваються вхідні", async ({ page }) => {
    await page.goto("https://example.com/sign-in");
    await page.getByLabel("Email").fill("reader@example.com");
    await page.getByRole("button", { name: "Увійти" }).click();
    await expect(page.getByRole("heading", { name: "Вхідні" })).toBeVisible();
});
```

Більшу частину роботи виконують два механізми. Перед дією Playwright робить перевірки готовності елемента (actionability) і чекає, доки всі вони пройдуть. Для кліку локатор має вказувати рівно на один елемент, а сам елемент має бути видимим, стабільним, тобто мати ту саму обмежувальну рамку щонайменше два кадри анімації поспіль, здатним отримати клік, а не лежати під іншим елементом, і активним. Якщо перевірки не проходять за відведений час, дія завершується помилкою `TimeoutError`. Твердження (assertions) у формі `await expect(locator)` поводяться так само: вони знову й знову отримують елемент, доки умова не справдиться або не мине тайм-аут твердження, за замовчуванням п’ять секунд.

Застосуйте поняття до рішення

Застосуйте «Що таке Playwright?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Одна команда веде ідею від ескізу до продукту, у який ваші клієнти вже можуть увійти, а не перекидає роботу між трьома підрядниками.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Збільшила дохід GIS-застосунку в 10 разів — 10x зростання виторгу завдяки картографічному продукту

Збільшила дохід GIS-застосунку в 10 разів Зв'язатися з нами →

Чому це важливо

Рішення, якому служить Playwright, стосується того, яку поведінку треба перевіряти в справжньому браузері і як зробити ці перевірки настільки надійними, щоб червоній збірці вірили. Деякі вади існують лише тоді, коли сторінку вже зібрано: діалог, що закриває кнопку, яку мав показати, макет, що ховає поле на певній ширині, скрипт, що працює в одному рушії й падає з помилкою в іншому. Тести, які запускають шматки коду окремо, їх не бачать.

Браузерні тести стають нестабільними (flaky), коли тест і сторінка змагаються наввипередки. Сторінка завантажує дані, анімується й перемальовується, а тест, що клікає тієї миті, коли виконується його рядок, іноді клікає зарано.

Давнім засобом була фіксована пауза: закоротка на повільній машині й марна трата часу на швидкій. Автоматичне очікування та твердження, що повторюють перевірку, замінюють паузу умовою, від якої тест насправді залежить. Посібник про твердження каже прямо: вебсторінки показують інформацію асинхронно, і твердження, які перевіряють лише раз, можуть зробити тест нестабільним.

Другий вибір — як тест знаходить елементи. Посібник про локатори радить насамперед атрибути, які бачить користувач, а понад усе локатори за роллю, бо `getByRole` з доступним ім'ям найближче до того, як сторінку сприймають люди й допоміжні технології. Це має корисний побічний ефект: тест, що натискає кнопку з назвою «Увійти», зламається, якщо кнопка втратить назву, а це та сама вада, на яку натрапить людина з екранним читачем. Посібник також каже, що локатори за роллю не замінюють аудитів доступності й тестів на відповідність. Ідентифікатори для тестів — запасний варіант. Посібник називає їх найстійкішим способом і водночас таким, якого не бачать користувачі, тож набір, що знаходить усе за id, доводить менше про те, що відчувають відвідувачі. CSS- і XPath-селектори не рекомендовані, бо DOM часто змінюється.

Ціна — час. Запуск браузера й завантаження сторінки роблять браузерні тести найповільнішим видом тестів у проєкті. Коли такий тест завершується з помилкою на сервері збірки, розібратися в причині допомагає trace viewer. Документований приклад конфігурації записує трасу під час першого повторного запуску тесту, що впав, з двома повторами на CI й жодним локально; вбудовані типові налаштування засобу запуску тестів не передбачають ані записування трас, ані повторних запусків. Траса проводить через кожну дію зі сторінкою до і після неї, мережевими запитами й консоллю.

Як це виглядає на практиці

Одна платформа тут тримає бізнес-логіку в базі даних, і роботою було побудувати багаторівневий набір тестів на чотирьох рівнях. Завданням було перевіряти поведінку кожного рівня там, де вона насправді живе, «а не перевіряти все ззовні через браузер». В API на Go — 981 тестова функція. У базі даних — 494 файли поведінкових тестів, тож збережену функцію тестують у самій базі даних, а не крізь три рівні застосунку над нею. У фронтенді — 543 специфікації Vitest і Playwright, зокрема 48 файлів наскрізних тестів і 30 браузерних специфікацій; сторінка дає одне спільне число для обох інструментів і не ділить його. Увесь набір запускається як 10

задач у workflow на 612 рядків, і сторінка прямо називає ціну: набір повільний, кожна зміна сплачує йому свою данину, а за такого розміру він і сам потребує догляду.

Цей сайт керує власними браузерними перевірками за допомогою іншої бібліотеки, Puppeteer, і сторінка подяк показує той самий компроміс. Ці перевірки охоплюють те, чого валідатор зробити не може, наприклад вимірювання контрасту, порядок фокуса, розмір цілі, вихід за межі вікна й розбиття на сторінки під час друку. Усі, крім виходу за межі вікна, входять до обов'язкових перевірок перед кожним пушем, а його, як і попіксельну візуальну регресію, людина запускає на вимогу.

Де виникають проблеми

Більшість проблем із Playwright походить від тестів, які перевіряють менше, ніж обіцяють їхні назви.

Фіксована пауза підміняє умову. Тест чекає дві секунди й потім читає сторінку або один раз читає значення й порівнює його звичайним твердженням. На завантаженому сервері збірки двох секунд іноді замало. Перевіряйте те, що побачили б відвідувачі, твердженням, яке повторює перевірку, наприклад `toHaveText`.

Елементи шукають за розміткою. Селектор крізь три вкладені елементи `div` або клас оформлення ламається з наступним редизайном, хоча сторінка й далі працює. Шукайте елемент за роллю й доступним ім'ям або за підписом чи текстом, а ідентифікатор для тестів додавайте лише там, де елемент неможливо впізнати за тим, що сприймають користувачі.

Успішний локатор за роллю вважають перевіркою доступності. `getByRole` доводить, що один елемент має ту роль і те ім'я, які задав тест. Про контраст, порядок фокуса чи решту сторінки він не каже нічого. Для цього запускайте з тесту рушій правил, наприклад axe, як це робить власний посібник Playwright через `@axe-core/playwright`; сканування охоплює сторінку в тому стані, у якому вона є на момент виклику, тож спершу відкрийте меню чи діалог.

Клік проштовхують силоміць. Параметр `force` вимикає неістотні перевірки готовності; для кліку Playwright перестає перевіряти, чи ціль справді отримує подію. Тест, якому він потрібен, часто обходить те, на що

натрапили б і відвідувачі, наприклад банер над кнопкою. Сприймайте початкову помилку як знахідку: або закрийте банер так, як це зробили б відвідувачі, або виправте перекриття.

Нестабільний тест перезапускають, доки він не пройде. Коли повтори ввімкнено, тест, що впав під час першого запуску й пройшов під час повтору, позначається як flaky, а не як невдалий. Тест, який лишається нестабільним тиждень у тиждень, або змагається зі сторінкою, або ловить справжню ваду, що то з'являється, то зникає, і траса, записана під час його першого повтору, — місце, з якого варто почати.

Усе перевіряють через браузер. Правило про ціни чи права доступу, перевірене кліками крізь три екрани, працює повільно й ламається з причин, які до правила не мають стосунку. Перевіряйте правило там, де воно живе, а браузерні тести залиште для того, що може показати лише браузер: шлях користувача, макет і відмінності між рушіями.

Що почитати далі

- Installation — встановлення Playwright Test, запуск прикладів тестів, HTML-звіт і UI Mode.
- Locators — рекомендований порядок пошуку елементів, від ролі й підпису до ідентифікаторів для тестів.
- Auto-waiting — перевірки готовності перед кожною дією і що саме вимикає `force` .
- Assertions — твердження з повторною перевіркою і без неї, тайм-аут за замовчуванням, м'які твердження й опитування.
- Trace viewer — запис траси під час повтору й покроковий перегляд тесту, що впав.
- Browsers — встановлення рушіїв, запуск Chrome і Edge та два режими headless.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Playwright?» стосується вашої організації.
Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/playwright/