Engineer

Playwright

værktøjet, der styrer en browser i tests

Hvad det er

Playwright er et open source-framework fra Microsoft til at automatisere webbrowsere med én programmeringsgrænseflade til Chromium, Firefox og WebKit. Playwright Test, udgivet på npm som `@playwright/test`, er den testkører, der er bygget oven på det, og det er den, de fleste mener, når de bruger navnet. Samme projekt udgiver også selve biblioteket til scripts, der ikke er test, et kommandolinjeværktøj og en MCP-server rettet mod AI-agenter samt en udvidelse til Visual Studio Code, alt sammen under Apache License 2.0.

Ud over JavaScript og TypeScript findes samme browserstyring til Python, hvor projektet anbefaler sit pytest-plugin, til Java med et vilkårligt testframework som JUnit og til .NET med basisklasser til MSTest, NUnit og xUnit.

`npx playwright install` henter Playwrights egne builds af de tre motorer, og fordi hver udgivelse flytter de versioner frem, kan kommandoen skulle køres igen efter en opgradering. Playwright kan også styre Google Chrome og Microsoft Edge, hvor de allerede er installeret på maskinen. Som standard kører test headless, uden et synligt vindue. Eksemplet i installationsvejledningen kører test parallelt i alle tre motorer via konfigurerede browserprojekter; uden disse projekter bruger testkørslen Chromium som standardbrowser. Hver test får en frisk browserkontekst, svarende til en ny browserprofil, så én tests cookies og lager ikke følger med over i den næste.

En test læses som de trin, et menneske ville tage:

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

test("login åbner indbakken", async ({ page }) => {
    await page.goto("https://example.com/sign-in");
```

```
    await page.getByLabel("E-mail").fill("reader@example.com");
    await page.getByRole("button", { name: "Log ind" }).click();
    await expect(page.getByRole("heading", { name: "Indbakke" })).toBeVisible();
});
```

To mekanismer bærer det meste af vægten. Før en handling kører Playwright en række tjek af, om elementet er klar til handlingen, på engelsk actionability, og venter, til de alle består. Ved et klik skal locatoren pege på præcis ét element, og elementet skal være synligt, stabilt, hvilket vil sige samme afgrænsningsboks i mindst to animationsbilleder i træk, i stand til at modtage klikket i stedet for at ligge under et andet element, og aktivt. Består tjekkene ikke inden for tidsgrænsen, ender handlingen i en `TimeoutError`. Assertions skrevet som `await expect(locator)` opfører sig på samme måde: de henter elementet igen og igen, indtil betingelsen holder, eller tidsgrænsen for assertions, som standard fem sekunder, løber ud.

Brug begrebet i en beslutning

Brug »Hvad er Playwright?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Ét team, der tager en idé fra skitse til noget, jeres kunder kan logge ind på, i stedet for at kaste den mellem tre leverandører.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har ført en GIS-kortapp til 10x omsætningsvækst — 10x omsætningsvækst, drevet af kortproduktet

Har ført en GIS-kortapp til 10x omsætningsvækst Kontakt os →

Hvorfor det betyder noget

Den beslutning, Playwright tjener, er, hvilken adfærd der skal tjekkes i en rigtig browser, og hvordan de tjek bliver troværdige nok til, at et rødt build bliver taget alvorligt. Nogle fejl findes kun, når siden er samlet: en dialog, der dækker den knap, den skulle vise frem, et layout, der skjuler et felt ved én bredde, et script, der virker i én motor og kaster en fejl i en anden. Test, der kører kodestykker isoleret, kan ikke se dem.

Browsertest bliver ustabile, på engelsk flaky, når testen og siden løber om kap. Siden henter data, animerer og renderer igen, og en test, der klikker i det øjeblik, dens linje kører, klikker nogle gange for tidligt. Det gamle middel var en fast pause, for kort på en langsom maskine og spildt tid på en hurtig. Automatisk ventetid og assertions, der prøver igen, erstatter pausen med den betingelse,

testen faktisk afhænger af. Vejledningen om assertions siger det ligeud: websider viser information asynkront, og assertions, der kun tjekker én gang, kan føre til ustabile test.

Det andet valg er, hvordan en test finder ting. Vejledningen om locators anbefaler brugervendte attributter først og rollelocators frem for alt, fordi `getByRole` med et tilgængeligt navn er det nærmeste, en test kommer på, hvordan brugere og hjælpemidler opfatter siden. Det har en nyttig sideeffekt: en test, der klikker på knappen med navnet »Log ind«, går i stykker, hvis knappen mister sit navn, og det er den samme fejl, en skærmlæserbruger ville møde. Vejledningen siger også, at rollelocators ikke erstatter tilgængelighedsrevisioner og konformitetstest. Test-id'er er reserven. Vejledningen kalder dem den mest robuste mulighed og samtidig ikke brugervendte, så en suite, der finder alt via id, beviser mindre om, hvad de besøgende oplever. CSS- og XPath-selektorer anbefales ikke, fordi DOM'en ofte ændrer sig.

Prisen er tid. At starte en browser og indlæse en side gør browsertest til den langsomste slags test, et projekt har. Fejler en af dem på en buildserver, hjælper trace viewer med at undersøge fejlen. Det dokumenterede konfigurationseksempel optager et spor, et trace, ved første genkørsel af en fejlet test, med genkørsler sat til to på CI og nul lokalt; testkørslens indbyggede standardindstillinger er hverken at optage spor eller at køre test igen. Sporet går gennem hver handling med siden før og efter, netværkskaldene og konsollen.

Sådan ser det ud i praksis

Én platform her holder sin forretningslogik i databasen, og arbejdet var at bygge en lagdelt testsuite på tværs af fire lag. Hvert lags adfærd skulle kontrolleres dér, hvor adfærden faktisk hører hjemme, »i stedet for at alt blev kontrolleret udefra gennem en browser«. Go-API'et har 981 testfunktioner. Databasen har 494 adfærdstestfiler, så en stored function bliver testet i databasen og ikke gennem tre applikationslag ovenover. Frontenden har 543 Vitest- og Playwright-specs, heraf 48 end-to-end-filer og 30 browser-specs; siden giver ét samlet tal for de to værktøjer og deler det ikke op. Hele suiten kører som 10 jobs i et workflow på 612 linjer, og siden siger ligeud, hvad det koster: suiten er langsom, den lægger en ekstra byrde på hver ændring, og i den størrelse skal den selv vedligeholdes.

Sitet her styrer sine egne browsertjek med et andet bibliotek, Puppeteer, og kolofonen viser samme afvejning. De tjek dækker det, en validator ikke kan afgøre, for eksempel målt kontrast, fokusrækkefølge, klikmålets størrelse,

overløb og sideskift på tryk. Porten kører dem alle undtagen overløbet før hvert push; det tjek kører et menneske efter behov, ligesom den pixelnøjagtige visuelle regression.

Hvor det går galt

De fleste problemer med Playwright kommer af test, der tjekker mindre, end navnet lover.

En fast pause står i stedet for en betingelse. En test venter to sekunder og læser så siden, eller den læser en værdi én gang og sammenligner den med en generisk assertion. På en travl buildserver er to sekunder nogle gange for kort. Tjek det, en besøgende ville se, med en web-first assertion som `toHaveText`, og lad den prøve igen.

Elementer bliver fundet ud fra deres markup. En selektor gennem tre indlejrede `div`-elementer eller en stilklasse går i stykker ved næste redesign, mens siden stadig virker. Find elementet ud fra rolle og tilgængeligt navn eller ud fra etiket eller tekst, og tilføj kun et test-id, hvor det ikke kan genkendes på noget, en bruger opfatter.

En rollelocator, der består, bliver taget som et tilgængelighedstjek. `getByRole` beviser, at ét element har den rolle og det navn, testen bad om. Om kontrast, fokusrækkefølge eller resten af siden siger den intet. Til det skal en regelmotor som axe køre fra testen, som Playwrights egen vejledning gør med `@axe-core/playwright`; scanningen dækker siden i den tilstand, den har, når man kalder den, så åbn menuen eller dialogen først.

Et klik bliver tvunget igennem. Indstillingen `force` slår de ikke-væsentlige tjek fra; ved et klik holder Playwright op med at tjekke, at målet faktisk modtager hændelsen. En test, der kræver den, arbejder ofte uden om noget, en besøgende også ville møde, som et banner over knappen. Behandl den oprindelige fejl som et fund, og luk enten banneret, som en besøgende ville, eller ret overlappet.

En ustabil test bliver kørt igen, til den består. Med genkørsler slået til bliver en test, der fejler i første kørsel og består ved en genkørsel, markeret som flaky og ikke som fejlet. En test, der er flaky uge efter uge, løber enten om kap med siden eller fanger en reel fejl, der kommer og går, og sporet fra dens første genkørsel er stedet at begynde.

Alt bliver tjekket gennem browseren. En regel om priser eller rettigheder, testet ved at klikke gennem tre skærmbilleder, er langsom og går i stykker af grunde, der intet har med reglen at gøre. Test reglen, hvor den bor, og hold browsertest til det, kun en browser kan vise: forløbet, layoutet og forskellene mellem motorerne.

Læs mere

- Installation — installation af Playwright Test, kørsel af eksempeltestene, HTML-rapporten og UI Mode.
- Locators — den anbefalede rækkefølge for at finde elementer, fra rolle og etiket til test-id'er.
- Auto-waiting — tjekkene bag hver handling, og hvad `force` slår fra.
- Assertions — assertions med og uden genforsøg, standardtidsgrænsen, bløde assertions og polling.
- Trace viewer — at optage et spor ved genkørsel og gå en fejlet test igennem handling for handling.
- Browsers — installation af motorerne, kørsel af Chrome og Edge og de to headless-tilstande.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Playwright?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/playwright/