Engineer

Playwright

verktøyet som styrer en nettleser i tester

Hva det er

Playwright er et rammeverk med åpen kildekode fra Microsoft for å automatisere nettlesere, med ett programmeringsgrensesnitt for Chromium, Firefox og WebKit. Playwright Test, publisert på npm som `@playwright/test`, er testkjøreren som er bygd oppå det, og det er den de fleste mener når de bruker navnet. Det samme prosjektet leverer det rene biblioteket for skript som ikke er tester, et kommandolinjeverktøy, en MCP-server rettet mot KI-agenter og en utvidelse for Visual Studio Code, alt under Apache License 2.0.

Utenom JavaScript og TypeScript er den samme styringen av nettleseren tilgjengelig fra Python, der prosjektet anbefaler sin egen pytest-plugin, fra Java, med et hvilket som helst testrammeverk som JUnit, og fra .NET, med basisklasser for MSTest, NUnit og xUnit.

`npx playwright install` laster ned prosjektets egne bygg av de tre motorene, og fordi hver utgivelse flytter de versjonene fremover, kan kommandoen måtte kjøres igjen etter en oppgradering. Playwright kan også styre Google Chrome og Microsoft Edge der de allerede finnes på maskinen. Som standard kjører testene hodeløst (headless), uten et synlig vindu. Eksemplet i installasjonsveiledningen kjører tester parallelt på alle tre motorene gjennom konfigurerte nettleserprosjekter; uten disse prosjektene er Chromium standardnettleseren i testkjøreren. Hver test får en fersk nettleserkontekst, det samme som en ny nettleserprofil, så informasjonskapsler og lagrede data fra én test følger ikke med til den neste.

En test leses som de stegene en person ville tatt:

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

test("signing in opens the inbox", async ({ page }) => {
    await page.goto("https://example.com/sign-in");
    await page.getByLabel("Email").fill("reader@example.com");
    await page.getByRole("button", { name: "Sign in" }).click();
    await expect(page.getByRole("heading", { name: "Inbox" })).toBeVisible();
});
```

To mekanismer bærer det meste av vekten. Før en handling kjører Playwright kontroller av om elementet er klart for den (actionability checks), og venter til alle er bestått. For et klikk må lokatoren (locator) peke på nøyaktig ett element, og det elementet må være synlig, stabilt, altså med den samme omsluttende boksen i minst to animasjonsrammer på rad, i stand til å ta imot klikket i stedet for å ligge under et annet element, og aktivert. Handlingen ender med en `TimeoutError` hvis kontrollene ikke er bestått innen tidsgrensen. Påstander skrevet som `await expect(locator)` oppfører seg på samme måte: de henter elementet igjen og igjen til betingelsen holder eller tidsgrensen for påstanden, fem sekunder som standard, går ut.

Bruk begrepet i en beslutning

Bruk «Hva er Playwright?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Ledet en GIS-kartapp til 10 ganger mer omsetning — 10x inntektsvekst, ledet fra kartproduktet

Ledet en GIS-kartapp til 10 ganger mer omsetning Ta kontakt →

Hvorfor det er viktig

Beslutningen Playwright hjelper med, er hvilken oppførsel som må kontrolleres i en ekte nettleser, og hvordan de kontrollene gjøres pålitelige nok til at et rødt bygg blir trodd. Noen feil finnes bare når siden er satt sammen: en dialog som dekker knappen den skulle vise frem, en layout som skjuler et felt ved én bredde, et skript som virker i én motor og kaster et unntak i en annen. Tester som kjører kodebiter hver for seg, ser dem aldri.

Nettlesertester blir ustabile når testen og siden kappløper. Siden henter data, animerer og tegner seg selv på nytt, og en test som klikker i det øyeblikket linjen kjører, klikker noen ganger for tidlig. Den vanlige utveien var en fast pause, for

kort på en treg maskin og bortkastet tid på en rask. Automatisk venting og påstander som prøver på nytt, erstatter pausen med den betingelsen testen faktisk er avhengig av. Veiledningen om påstander sier det rett ut: nettsider viser informasjon asynkront, og påstander som sjekker bare én gang, gir ustabile tester.

Det andre valget er hvordan en test finner ting. Veiledningen om lokatorer anbefaler attributter brukeren selv ser, først og fremst rollelokatorer, fordi `getByRole` med et tilgjengelig navn (accessible name) er det nærmeste en test kommer måten brukere og skjermlesere og andre hjelpemidler oppfatter siden på. Det har en bivirkning som er verdt å ha: en test som klikker på knappen med navnet «Sign in», brekker hvis knappen mister navnet sitt, og det er den samme feilen en som bruker skjermleser, ville møtt. Veiledningen sier også at rollelokatorer «ikke erstatter revisjoner av universell utforming og samsvarstester». Test-ID-er er reserveløsningen. Veiledningen kaller dem det mest robuste valget og samtidig «ikke synlige for brukeren», så en testsuite som finner alt via ID-er, beviser mindre om hva en besøkende opplever. CSS- og XPath-selektorer anbefales ikke, fordi DOM-en endres ofte.

Prisen er tid. Å starte en nettleser og laste en side gjør nettlesertester til den tregeste typen tester et prosjekt har. Når én av dem feiler på en byggserver, er sporingsvisningen (trace viewer) utgangspunktet for feilsøkingen. Det dokumenterte konfigurasjonseksemplet tar opp en sporing ved første nye forsøk på en mislykket test, med to nye forsøk i CI og null lokalt; testkjørerens innebygde standardinnstillinger gir verken sporing eller nye forsøk. Sporingen går gjennom hver handling med siden før og etter den, nettverksforespørslene og konsollen.

Slik ser det ut i praksis

En plattform her har forretningslogikken sin i databasen, og arbeidet var å bygge en lagdelt testsuite over fire lag. Oppgaven var å kontrollere oppførselen til hvert lag der oppførselen faktisk hører hjemme, «i stedet for at alt ble kontrollert utenfra gjennom en nettleser». Go-API-et har 981 testfunksjoner. Databasen har 494 atferdstestfiler, så en lagret funksjon testes i databasen i stedet for gjennom tre lag med applikasjon over den. Frontenden har 543 Vitest- og Playwright-spesifikasjoner, blant dem 48 ende-til-ende-filer og 30 nettleserspesifikasjoner; siden oppgir ett samlet tall for begge verktøyene. Hele testsuiten kjører som 10 jobber i en arbeidsflyt på 612 linjer, og siden er direkte om prisen: testsuiten er treg, den legger en avgift på hver endring, og i denne størrelsen trenger den eget vedlikehold.

Dette nettstedet styrer sine egne nettleserkontroller med et annet bibliotek, Puppeteer, og kolofonen viser den samme avveiningen. Kontrollene dekker det en validator ikke kan avgjøre, som målt kontrast, fokusrekkefølge, størrelse på klikkflater, overløp i visningsområdet og sideinndeling ved utskrift. Kvalitetsporten kjører alle unntatt overløpet før hver push; den sjekken kjører et menneske ved behov, akkurat som den visuelle regresjonstesten piksel for piksel.

Hvor det går galt

Det meste av trøbbelet med Playwright kommer av tester som sjekker mindre enn navnene deres lover.

En fast pause står i stedet for en betingelse. En test venter to sekunder og leser så siden, eller leser en verdi én gang og sammenligner den med en generell påstand. På en travel byggserver er to sekunder av og til for kort. Gjør påstanden om det en besøkende ville sett, med en web-first-påstand som `toHaveText`, og la den prøve på nytt.

Elementer blir funnet via markeringen. En selektor gjennom tre nøstede `div`-elementer eller en stilklasse brekker ved neste redesign, mens siden fortsatt virker. Finn elementet via rolle og tilgjengelig navn, eller via etikett eller tekst, og legg bare til en test-ID der ingenting en bruker oppfatter, identifiserer det.

En rollelokator som består, blir tatt for en kontroll av universell utforming. `getByRole` beviser at ett element har den rollen og det navnet testen spurte etter. Om kontrast, fokusrekkefølge og resten av siden sier den ingenting. Til det kjører du en regelmotor som axe fra testen, slik Playwrights egen veiledning gjør via `@axe-core/playwright`; skanningen dekker siden «slik den er i det øyeblikket du kaller den», så åpne menyen eller dialogen først.

Et klikk blir tvunget gjennom. Valget `force` slår av de kontrollene som ikke er helt nødvendige; for et klikk slutter Playwright å sjekke at målet faktisk tar imot hendelsen. En test som trenger det, jobber seg ofte rundt noe en besøkende også ville støtt på, som et banner over knappen. Behandle den opprinnelige feilen som et funn, og lukk enten banneret slik en besøkende ville gjort, eller rett overlappingen.

En ustabil test blir kjørt på nytt til den består. Med nye forsøk slått på blir en test som feiler første gang og består ved et nytt forsøk, merket som flaky, ustabil, i stedet for mislykket. En test som er ustabil uke etter uke, kappløper enten med siden eller fanger en ekte feil som kommer og går, og sporingen som ble tatt opp ved det første nye forsøket, er stedet å begynne å lete.

Alt blir kontrollert gjennom nettleseren. En regel om priser eller tilganger, testet ved å klikke seg gjennom tre skjermbilder, er treg og brekker av grunner som har lite med regelen å gjøre. Test regelen der den bor, og spar nettlesertestene til det bare en nettleser kan vise: brukerreisen, layouten og forskjellene mellom motorene.

Les mer

- Installation — slik installerer du Playwright Test, kjører eksempeltestene og bruker HTML-rapporten og UI Mode.
- Locators — den anbefalte rekkefølgen for å finne elementer, fra rolle og etikett til test-ID-er.
- Auto-waiting — kontrollene som kjøres før hver handling, og hva `force` slår av.
- Assertions — påstander som prøver på nytt og påstander som ikke gjør det, standard tidsgrense, myke påstander og polling.
- Trace viewer — å ta opp en sporing ved nytt forsøk og gå gjennom en mislykket test handling for handling.
- Browsers — installasjon av motorene, kjøring av merkevareversjonene av Chrome og Edge, og de to hodeløse modusene.



Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Playwright?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/playwright/