Engineer

Playwright

Outil qui pilote un navigateur pendant les tests

Ce que c’est

Playwright est un framework open source de Microsoft qui automatise les navigateurs web avec une interface de programmation commune à Chromium, Firefox et WebKit. Playwright Test, publié sur npm sous le nom `@playwright/test`, est l’exécuteur de tests construit sur cette base. C’est généralement lui que désigne le nom. Le même projet fournit aussi la bibliothèque seule pour des scripts qui ne sont pas des tests, un outil en ligne de commande, un serveur MCP destiné aux agents d’IA et une extension pour Visual Studio Code, le tout sous Apache License 2.0.

Outre JavaScript et TypeScript, ce pilotage des navigateurs est disponible en Python, pour lequel le projet recommande son plugin pytest, en Java avec tout framework de test tel que JUnit, et en .NET avec des classes de base pour MSTest, NUnit et xUnit.

`npx playwright install` télécharge les versions des trois moteurs préparées par le projet. Chaque nouvelle version faisant évoluer celles des moteurs, il peut être nécessaire de relancer la commande après une mise à niveau. Playwright peut aussi piloter Google Chrome et Microsoft Edge lorsqu’ils sont déjà installés. Sans autre configuration, les tests s’exécutent sans fenêtre visible, en mode headless. L’exemple du guide d’installation les exécute en parallèle sur les trois moteurs au moyen de projets de navigateur configurés ; sans ces projets, le navigateur présélectionné du lanceur de tests est Chromium. Chaque test reçoit un contexte de navigateur neuf, l’équivalent d’un nouveau profil : les cookies et le stockage d’un test ne passent donc pas au suivant.

Un test se lit comme les étapes suivies par une personne :

```
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();
});
```

Deux mécanismes assurent l'essentiel du travail. Avant une action, Playwright vérifie qu'elle est réalisable et attend que tous les contrôles réussissent. Pour un clic, le localisateur doit désigner exactement un élément. Celui-ci doit être visible, stable — son rectangle englobant doit rester identique pendant au moins deux images d'animation consécutives —, capable de recevoir le clic plutôt que recouvert par un autre élément, et activé. Si les contrôles ne réussissent pas avant l'expiration du délai, l'action se termine par une `TimeoutError`. Les assertions écrites avec `await expect(locator)` fonctionnent de la même manière : elles recherchent à nouveau l'élément jusqu'à ce que la condition soit remplie ou que le délai de l'assertion, de cinq secondes sans autre réglage, expire.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Playwright ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une seule équipe transforme une idée en un produit auquel vos clients peuvent se connecter, sans le faire passer entre trois prestataires.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Application GIS : revenus multipliés par 10 — 10x croissance du chiffre d'affaires portée par le produit cartographique

Application GIS : revenus multipliés par 10 Contactez-nous →

Pourquoi c'est important

Playwright aide à décider quels comportements doivent être vérifiés dans un vrai navigateur et comment rendre ces contrôles assez fiables pour qu'un build en échec soit pris au sérieux. Certains défauts n'apparaissent qu'une fois la page assemblée : une boîte de dialogue recouvre le bouton qu'elle devrait révéler, une disposition masque un champ à une certaine largeur, un script fonctionne dans un moteur mais lève une exception dans un autre. Des tests de fragments de code isolés ne peuvent pas les voir.

Les tests de navigateur deviennent intermittents lorsque le test et la page se livrent une course. La page charge des données, anime ses éléments et se réaffiche ; un test qui clique dès que sa ligne s'exécute clique parfois trop tôt. Le remède habituel était une pause fixe, trop courte sur une machine lente et inutilement longue sur une machine rapide. L'attente automatique et les assertions réessayées remplacent cette pause par la condition dont le test dépend réellement. Le guide des assertions le dit clairement : les pages affichent leurs informations de manière asynchrone, et les assertions qui ne vérifient qu'une fois rendent les tests intermittents.

Le deuxième choix concerne la manière de trouver les éléments. Le guide des localisateurs recommande d'abord les attributs perceptibles par les utilisateurs, et surtout les localisateurs par rôle : `getByRole` avec un nom accessible est ce qui se rapproche le plus de « la manière dont les utilisateurs et les technologies d'assistance perçoivent la page ». Cela présente un avantage supplémentaire : un test qui clique sur le bouton nommé « Sign in » échoue si ce bouton perd son nom, ce qui est aussi un défaut pour une personne utilisant un lecteur d'écran. Le guide précise que les localisateurs par rôle « ne remplacent pas les audits d'accessibilité et les tests de conformité ». Les identifiants de test sont le recours suivant. Le guide les décrit comme l'option la plus robuste, mais aussi comme « non destinés aux utilisateurs » : une suite qui ne trouve les éléments que par leur identifiant prouve moins de choses sur l'expérience des visiteurs. Les sélecteurs CSS et XPath sont déconseillés, car le DOM change souvent.

Le coût est le temps. Démarrer un navigateur et charger une page fait de ces tests les plus lents d'un projet. Lorsqu'un test échoue sur un serveur d'exécution des builds, la visionneuse de traces permet de reprendre l'enquête. L'exemple de configuration présenté dans la documentation enregistre une trace lors de la première nouvelle tentative d'un test en échec, avec deux nouvelles tentatives en CI et aucune en local ; les réglages intégrés par défaut du lanceur de tests désactivent le traçage et les nouvelles tentatives. La trace parcourt chaque action, avec la page avant et après, les requêtes réseau et la console.

À quoi cela ressemble en pratique

Une plateforme présentée ici conserve sa logique métier dans la base de données. Il fallait y construire une suite de tests répartie sur quatre couches. L'objectif annoncé était de vérifier le comportement de chaque couche là où il se trouve réellement, « plutôt que de tout vérifier de l'extérieur à travers un navigateur ». L'API Go compte 981 fonctions de test. La base compte 494 fichiers de tests comportementaux : une fonction stockée est donc testée dans la base

plutôt qu'à travers trois couches applicatives au-dessus d'elle. Le frontend compte 543 spécifications de test Vitest et Playwright, dont 48 fichiers de tests de bout en bout et 30 spécifications de test de navigateur ; la page donne un total commun aux deux outils. L'ensemble s'exécute en 10 tâches dans un flux de travail de 612 lignes. La page est explicite sur le prix : la suite est lente, pèse sur chaque changement et, à cette taille, exige sa propre maintenance.

Ce site pilote ses propres contrôles de navigateur avec une autre bibliothèque, Puppeteer, et sa page de crédits expose le même compromis. Ces contrôles couvrent ce qu'un validateur ne peut pas vérifier, comme le contraste mesuré, l'ordre de focus, la taille des cibles, les débordements de la zone d'affichage et la pagination imprimée. Le contrôle qualité obligatoire les exécute tous avant chaque envoi vers le dépôt distant, sauf celui des débordements. Ce dernier, comme le contrôle de régression visuelle pixel par pixel, attend qu'une personne le lance à la demande.

Où les problèmes surviennent

La plupart des difficultés avec Playwright viennent de tests qui vérifient moins que ce que leur nom annonce.

Une pause fixe remplace une condition. Un test attend deux secondes puis lit la page, ou lit une valeur une seule fois et la compare avec une assertion générique. Sur un serveur d'exécution des builds chargé, deux secondes sont parfois insuffisantes. Vérifiez ce qu'un visiteur verrait avec une assertion adaptée au web, comme `toHaveText`, et laissez-la réessayer.

Les éléments sont trouvés par leur balisage. Un sélecteur qui traverse trois éléments `div` imbriqués ou une classe de style échoue à la prochaine refonte, alors que la page fonctionne encore. Trouvez l'élément par son rôle et son nom accessible, son libellé ou son texte. N'ajoutez un identifiant de test que si rien de perceptible par l'utilisateur ne permet de l'identifier.

La réussite d'un localisateur par rôle est prise pour un contrôle d'accessibilité. `getByRole` prouve qu'un élément possède le rôle et le nom demandés par le test. Il ne dit rien du contraste, de l'ordre de focus ni du reste de la page. Pour cela, lancez depuis le test un moteur de règles comme axe, ainsi que le fait le guide de Playwright avec `@axe-core/playwright`. L'analyse couvre la page « dans son état au moment de l'appel » : ouvrez donc d'abord le menu ou la boîte de dialogue.

Un clic est forcé. L'option `force` désactive les contrôles non essentiels de faisabilité de l'action. Pour un clic, Playwright ne vérifie plus que la cible reçoit effectivement l'événement. Un test qui en a besoin contourne souvent un problème qu'un visiteur rencontrerait aussi, comme une bannière devant le bouton. Considérez l'erreur initiale comme un constat : fermez la bannière comme le ferait un visiteur ou corrigez le chevauchement.

Un test intermittent est réessayé jusqu'à réussir. Lorsque les nouvelles tentatives sont activées, un test qui échoue à la première exécution et réussit lors d'une nouvelle tentative est marqué comme intermittent plutôt qu'en échec. S'il reste intermittent semaine après semaine, il dépend d'une condition de concurrence avec la page ou révèle un véritable défaut occasionnel. La trace enregistrée lors de sa première nouvelle tentative est le point de départ de l'enquête.

Tout est vérifié à travers le navigateur. Vérifier une règle de prix ou de permissions en cliquant dans trois écrans est lent, et ce test échoue pour des raisons sans rapport avec la règle. Testez la règle là où elle se trouve et réservez les tests de navigateur à ce que seul un navigateur peut montrer : le parcours, la disposition et les différences entre moteurs.

Pour aller plus loin

- Installation — Installation de Playwright Test, exécution des exemples, rapport HTML et mode UI.
- Localisateurs — Ordre recommandé pour trouver les éléments, du rôle et du libellé aux identifiants de test.
- Attente automatique — Contrôles de faisabilité de chaque action et contrôles désactivés par `force` .
- Assertions — Assertions avec ou sans nouvelles tentatives, délai par défaut, assertions non bloquantes et interrogation périodique.
- Visionneuse de traces — Enregistrement d'une trace lors d'une nouvelle tentative et parcours d'un test en échec, action par action.
- Navigateurs — Installation des moteurs, utilisation des versions de Chrome et Edge distribuées par leurs éditeurs, et deux modes headless.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que Playwright ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/playwright/