Engineer

Playwright

Herramienta que controla un navegador en las pruebas

Qué es

Playwright es un framework de código abierto de Microsoft para automatizar navegadores web, con una misma interfaz de programación para Chromium, Firefox y WebKit. Playwright Test, publicado en npm como `@playwright/test`, es el ejecutor de pruebas construido sobre él y aquello a lo que se refiere la mayoría de las personas al usar el nombre. El mismo proyecto distribuye la biblioteca independiente para scripts que no son pruebas, una herramienta de línea de comandos, un servidor MCP dirigido a agentes de IA y una extensión para Visual Studio Code, todo ello bajo la licencia Apache 2.0.

Además de JavaScript y TypeScript, el mismo control del navegador está disponible desde Python, para el que el proyecto recomienda su complemento de pytest; desde Java, con cualquier framework de pruebas, como JUnit; y desde .NET, con clases base para MSTest, NUnit y xUnit.

`npx playwright install` descarga las compilaciones propias de los tres motores y, como cada versión incorpora versiones más recientes de estos, puede ser necesario repetir el comando después de una actualización. También puede controlar Google Chrome y Microsoft Edge si ya están instalados en la máquina. De forma predeterminada, las pruebas se ejecutan sin ventana visible. El ejemplo de la guía de instalación las ejecuta en paralelo en los tres motores mediante proyectos de navegador configurados; sin esos proyectos, el navegador predeterminado del ejecutor de pruebas es Chromium. Cada prueba recibe un contexto de navegador nuevo, equivalente a un perfil de navegador nuevo, para que las cookies y el almacenamiento de una prueba no pasen a la siguiente.

Una prueba se lee como los pasos que seguiría una persona:

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

Dos mecanismos hacen la mayor parte del trabajo. Antes de una acción, Playwright comprueba si puede realizarse y espera a que se cumplan todas las condiciones. Para un clic, el localizador debe identificar exactamente un elemento, y este debe ser visible, estable —es decir, su rectángulo delimitador debe permanecer igual durante al menos dos fotogramas de animación consecutivos—, capaz de recibir el clic en lugar de quedar debajo de otro elemento, y estar habilitado. Si las comprobaciones no se superan dentro del tiempo de espera, la acción termina con un `TimeoutError`. Las aserciones escritas como `await expect(locator)` funcionan de la misma manera: vuelven a obtener el elemento una y otra vez hasta que se cumple la condición o se agota el tiempo de espera de la aserción, fijado de forma predeterminada en cinco segundos.

Utilice el concepto en una decisión

Aplique «¿Qué es Playwright?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Un equipo que lleva una idea del boceto a un producto al que sus clientes pueden acceder, sin pasarla entre tres proveedores.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Ingresos multiplicados por 10 con mapas GIS — 10x crecimiento de ingresos impulsado desde el producto cartográfico

Ingresos multiplicados por 10 con mapas GIS Póngase en contacto →

Por qué importa

Playwright ayuda a decidir qué comportamiento hay que comprobar en un navegador real y cómo hacer que esas comprobaciones sean lo bastante fiables para que se confíe en un resultado fallido de la construcción. Algunos fallos solo existen cuando la página está montada: un diálogo que tapa el botón que debería mostrar, un diseño que oculta un campo con una anchura concreta o un script que funciona en un motor y lanza una excepción en otro. Las pruebas que ejecutan fragmentos de código de forma aislada no pueden verlos.

Las pruebas de navegador se vuelven inestables cuando la prueba y la página compiten por adelantarse. La página obtiene datos, anima elementos y vuelve a renderizarse, y una prueba que hace clic en cuanto se ejecuta su línea a veces lo hace demasiado pronto. El remedio habitual era una pausa fija, demasiado corta en una máquina lenta y una pérdida de tiempo en una rápida. La espera automática y las aserciones con reintentos sustituyen la pausa por la condición de la que depende realmente la prueba. La guía de aserciones lo dice claramente: las páginas web muestran información de forma asíncrona y las aserciones que comprueban una sola vez producen pruebas inestables.

La segunda decisión es cómo encuentra los elementos una prueba. La guía de localizadores recomienda dar prioridad a los atributos que percibe el usuario y, sobre todo, a los localizadores por rol, porque `getByRole` con un nombre accesible es lo más cercano a «cómo perciben la página los usuarios y las tecnologías de asistencia». Eso tiene un efecto secundario valioso: una prueba que hace clic en el botón llamado «Sign in» falla si el botón pierde su nombre, el mismo defecto que encontraría una persona que usa un lector de pantalla. La guía también indica que los localizadores por rol «no sustituyen las auditorías de accesibilidad ni las pruebas de conformidad». Los identificadores de prueba son la alternativa de reserva. La guía los considera la opción más resistente y también señala que «no están orientados al usuario», de modo que una batería que encuentra todo por identificador demuestra menos sobre la experiencia del visitante. No se recomiendan los selectores CSS y XPath, porque el DOM cambia a menudo.

El coste es tiempo. Iniciar un navegador y cargar una página convierte las pruebas de navegador en las más lentas de un proyecto. Cuando una falla en un servidor de construcción, el visor de trazas permite reconstruir lo sucedido. El ejemplo de configuración documentado registra una traza en el primer reintento de una prueba fallida, con dos reintentos en integración continua (CI) y cero en local; los valores predeterminados del propio ejecutor no activan el registro de trazas ni los reintentos. La traza recorre cada acción con la página antes y después, las peticiones de red y la consola.

Cómo se aplica en la práctica

Una de las plataformas de este trabajo mantiene su lógica de negocio en la base de datos, y el trabajo consistió en construir una batería de pruebas distribuida en cuatro capas. La tarea declarada era comprobar el comportamiento de cada capa allí donde realmente reside, «en lugar de comprobarlo todo desde fuera a través de un navegador». La API de Go contiene 981 funciones de prueba. La base de

datos contiene 494 archivos de pruebas de comportamiento, de modo que una función almacenada se prueba en la base de datos y no a través de tres capas de aplicación situadas encima. El frontend contiene 543 especificaciones de Vitest y Playwright, entre ellas 48 archivos de pruebas de extremo a extremo y 30 especificaciones de navegador; la página ofrece un único recuento para ambas herramientas. Toda la batería se ejecuta como 10 trabajos en un flujo de 612 líneas, y la página expone con claridad el coste: la batería es lenta, añade trabajo a cada cambio y, a este tamaño, necesita su propio mantenimiento.

Este sitio realiza sus comprobaciones de navegador con otra biblioteca, Puppeteer, y su página de créditos muestra el mismo equilibrio entre ventajas y costes. Esas comprobaciones cubren lo que no puede comprobar un validador, como el contraste medido, el orden del foco, el tamaño de las zonas interactivas, el desbordamiento del área de visualización y la paginación de impresión. El control de calidad obligatorio ejecuta todas ellas antes de cada envío al repositorio remoto, salvo la del desbordamiento del área de visualización; esta, al igual que la comparación de regresión visual píxel a píxel, espera a que alguien la ejecute expresamente.

Dónde surgen los problemas

La mayoría de los problemas con Playwright proceden de pruebas que comprueban menos de lo que anuncian sus nombres.

Una pausa fija sustituye a una condición. Una prueba espera dos segundos y después lee la página, o lee un valor una sola vez y lo compara con una aserción genérica. En un servidor de construcción ocupado, dos segundos a veces no bastan. Compruebe lo que vería un visitante con una aserción orientada a la web como `toHaveText` y deje que reintente la comprobación.

Los elementos se buscan por su marcado. Un selector que atraviesa tres elementos `div` anidados o una clase de estilo se rompe en el siguiente rediseño aunque la página siga funcionando. Localice el elemento por su rol y nombre accesible, por su etiqueta o por su texto, y añada un identificador de prueba solo cuando nada de lo que percibe el usuario lo identifique.

Un resultado correcto de un localizador por rol se toma por una comprobación de accesibilidad. `getByRole` demuestra que un elemento tiene el rol y el nombre solicitados por la prueba. No dice nada sobre el contraste, el orden del foco ni el resto de la página. Para ello, ejecute desde la prueba un

motor de reglas como axe, tal como hace la propia guía de Playwright mediante `@axe-core/playwright` ; el análisis cubre la página «en su estado actual en el momento de la llamada», por lo que antes hay que abrir el menú o el diálogo.

Se fuerza un clic. La opción `force` desactiva las comprobaciones no esenciales sobre la posibilidad de realizar la acción; para un clic, Playwright deja de comprobar que el objetivo recibe realmente el evento. Una prueba que la necesita suele estar esquivando algo que también encontraría un visitante, como un banner encima del botón. Trate el error original como un hallazgo y cierre el banner como lo haría un visitante o corrija el solapamiento.

Una prueba inestable se reintenta hasta que pasa. Con los reintentos activados, una prueba que falla en la primera ejecución y pasa en un reintento se marca como inestable en lugar de fallida. Si lo hace semana tras semana, sufre una condición de carrera con la página o detecta un fallo intermitente real. La traza registrada en el primer reintento es el punto por el que empezar a investigar.

Todo se comprueba a través del navegador. Comprobar una regla sobre precios o permisos haciendo clic en tres pantallas es lento, y esa prueba falla por motivos ajenos a la regla. Pruebe la regla donde reside y reserve las pruebas de navegador para lo que solo puede mostrar un navegador: el recorrido, el diseño y las diferencias entre motores.

Lecturas adicionales

- Instalación — Instalar Playwright Test, ejecutar las pruebas de ejemplo, el informe HTML y el modo de interfaz de usuario.
- Localizadores — Orden recomendado para encontrar elementos, desde rol y etiqueta hasta identificadores de prueba.
- Espera automática — Comprobaciones previas a cada acción y cuáles desactiva `force` .
- Aserciones — Aserciones con y sin reintentos, tiempo de espera predeterminado, aserciones que no interrumpen la prueba y sondeo.
- Visor de trazas — Registrar una traza en el reintento y recorrer una prueba fallida acción por acción.
- Navegadores — Instalar los motores, ejecutar Chrome y Edge de sus respectivos proveedores y los dos modos sin ventana.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Playwright?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/playwright/