**<https://www.browserstack.com/guide/playwright-tutorial>**

**Empezando Playwright**

¿Qué es el dramaturgo?

[Playwright](https://playwright.dev/) es una biblioteca de automatización de pruebas de código abierto desarrollada inicialmente por colaboradores de Microsoft. Admite lenguajes de programación como Java, Python, C# y NodeJS. Playwright viene con licencia Apache 2.0 y es más popular con NodeJS con Javascript/Typecript. Este tutorial de Playwright le ayudará a configurar NodeJS utilizando Visual Studio Code.

¿Por qué elegir Playwright Automation?

Aunque Playwright es nuevo en el mercado, difícilmente podemos enumerar las limitaciones, ya que admite varios idiomas. Las personas que quieran migrar de Selenium a Playwright pueden hacerlo rápidamente ya que Playwright admite C#, Java y Python. Los lenguajes de programación no son una barrera. El primer lanzamiento de Playwright fue en enero de 2020 y desde entonces ha ganado mucha popularidad.

* Aprovecha el protocolo DevTools para escribir pruebas automatizadas potentes y estables.
* Puede ver y controlar el navegador en lugar de depender de una capa de traducción intermedia; permite la simulación de escenarios de usuario más reveladores y relevantes.

**Ventajas de la automatización del dramaturgo**

Los colaboradores de los dramaturgos son muy activos en el lanzamiento de nuevas funciones cada mes, que se enumeran a continuación:

1. **Fácil instalación y configuración:**al ser un [marco de automatización de pruebas](https://www.browserstack.com/guide/best-test-automation-frameworks) , solo necesita una configuración ya que la instalación no lleva mucho tiempo. Dependiendo del idioma que usemos con Playwright, los pasos de instalación pueden cambiar
2. **Compatibilidad con varios navegadores:**todos los navegadores de la familia Chromium (Chrome, Edge), Webkit (Safari) y Firefox son compatibles.
3. **Compatibilidad con varios idiomas:**Playwright admite Java, C#, Python, Javascript/Typecript, lo que la convierte en una opción popular. La mayoría de los marcos modernos de automatización de pruebas de código abierto omiten esta característica.
4. **Tipos de pruebas:**Playwright admite pruebas funcionales, de extremo a extremo y de API. Con un complemento de terceros, Playwright se puede integrar con [Accessibility Testing](https://www.browserstack.com/accessibility-testing" \o "Pruebas de accesibilidad" \t "_blank) .
5. **Pruebas de navegador paralelo:** Playwright también admite la ejecución de pruebas simultáneas (también conocidas como  [pruebas paralelas](https://www.browserstack.com/guide/what-is-parallel-testing) ) a través *del contexto del navegador* y puede ejecutar pruebas paralelas con varios navegadores. Esto amplía las pruebas y resulta útil cuando se deben probar varias páginas web simultáneamente.
6. **Compatibilidad con múltiples pestañas/ventanas del navegador:** Playwright admite *múltiples pestañas y múltiples ventanas* . Algunos casos de prueba deben verificar el escenario iniciando una nueva ventana y regresando a la ventana principal. Playwright admite todos los diferentes tipos de casos de prueba.
7. **Reporteros integrados:**el marco Playwright, de forma predeterminada, viene con muchos reporteros valiosos como List, Dot, Line, JSON, JUnit y HTML Reporters. Lo interesante es que con Playwright, uno puede crear reporteros personalizados. Playwright también apoya al reportero externo Allure Report.
8. **Compatibilidad con mecanografiado lista para usar:**no se requiere configuración para la compatibilidad con el lenguaje mecanografiado, ya que comprende su código mecanografiado y javascript.
9. **Soporte de integración CI/CD:**Playwright admite la integración CI/CD. Incluso proporciona imágenes acoplables para algunos enlaces de idiomas.
10. **Compatibilidad con herramientas de depuración:**las pruebas de Playwright admiten diferentes opciones de depuración, lo que las hace fáciles de usar para los desarrolladores. Algunas opciones de depuración son Playwright Inspector, VSCode Debugger, Browser Developer Tools y Trace Viewers Console Logs.

**Otras características notables de Playwright incluyen** :

* Soporte de marco flotante
* [Soporte para el modelo de objetos de página](https://www.browserstack.com/guide/page-object-model-with-playwright)
* Reporteros incorporados
* Patrón de objeto de página
* Soporte de navegación entre orígenes
* Soporte de selectores
* DOM en la sombra
* Espera automática
* Soportes de ejecución de pruebas de terceros
* Vídeos y captura de pantalla
* Emulación del navegador
* reintento de prueba,
* Proyecto parametrizado, etc.

**Leer más:**[Cómo ejecutar sitios web locales usando Playwright](https://www.browserstack.com/docs/automate/playwright/local-testing)

**Limitaciones de la automatización de dramaturgos**

1. El dramaturgo es nuevo y todavía está evolucionando: hay margen de mejora.
2. Sin soporte para IE11
3. Playwright no admite aplicaciones móviles nativas
4. Aunque la documentación de Playwright es excelente, no podemos esperar mucho en cuanto al apoyo de la comunidad. Dado que Playwright es nuevo en el mercado, la base de usuarios sigue creciendo.

**Primeros pasos con la configuración de Playwright**

Este tutorial de Playwright explica cómo configurar NodeJS fácilmente.

**Requisitos previos:**

1. **Instale Visual Studio Code:**[descargue](https://code.visualstudio.com/) e instale Visual Studio Code (VSCode).
2. **Instalar NodeJS:**[Descargar](https://nodejs.org/en/) e instalar Node JS

**Cómo instalar y ejecutar el script de prueba de Playwright**

**Paso 1:** cree un directorio nuevo *(por ejemplo, PlaywrightDemo)* en VSCode

**Paso 2:** abra el directorio en Visual Studio Code. Desde el código VS,

*Haga clic en Archivo > Abrir carpeta > Elegir carpeta recién creada (PlaywrightDemo)*

**Paso 3:** desde VS Code, *haga clic en Menú Terminal > Haga clic en Nueva Terminal*

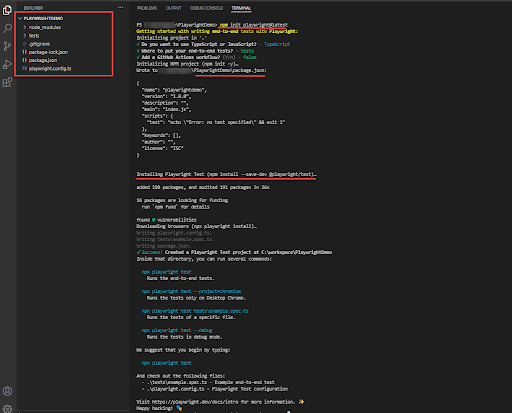
**Paso 4:** Ingrese el siguiente comando para iniciar la instalación de Playwright

npm init dramaturgo@último

***Nota:****El comando anterior plantea una serie de preguntas. Por favor proporcione los aportes apropiados. En este tutorial, utilizamos****lenguaje mecanografiado.***

El comando anterior realiza la siguiente operación:

* Crea paquete.json
* Instala la biblioteca npm
* Configura archivos y carpetas básicos
  + **Carpeta de pruebas:**esta carpeta contiene scripts de prueba reales. De forma predeterminada, se creará un archivo example.spec.ts dentro de esta carpeta.
  + **.gitignore:**este archivo ayuda si está utilizando el repositorio git
  + **package.json y package-lock.json:**este archivo ayuda a rastrear dependencias, crear un acceso directo para ejecutar pruebas, etc.
  + **playwright.config.ts:**este es el archivo de configuración global de Playwright, que puede configurar con las opciones disponibles.



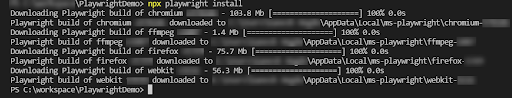
**Paso 5: instalar navegadores**

Sin embargo, Playwright está configurado para ejecutarse en navegadores existentes, lo que puede crear problemas al ejecutar las pruebas, por lo que se recomienda utilizar los navegadores Playwright. Usando el siguiente comando, puede instalar todos los navegadores diferentes en Playwright.

Code:

instalación de dramaturgo npx

El comando anterior instala la versión Playwright de los navegadores Chromium, Firefox y Webkit.



**Paso 6: crea la primera prueba de dramaturgo**

Navegue dentro de la carpeta de pruebas y cree un archivo de especificaciones de prueba, por ejemplo: **demo.spec.ts**

Comencemos un caso de prueba con el siguiente escenario.

**Guión:**

* Navegue a <https://www.browserstack.com/>
* Haga clic en Iniciar sesión
* Ingrese credenciales incorrectas
* Verifique que el mensaje de error se muestre o no.

El **demo.spec.ts**es nuestro script de prueba de Playwright de la siguiente manera

//demo.spec.ts

importar { prueba , esperar , página } desde '@ dramaturgo/prueba' ;

prueba . antes de cada ( async ({ página }) => {

  espera la página . ir a ( 'https://www.browserstack.com/' );

});

prueba . describir ( 'Prueba de demostración' , () => {

    test ( 'Verificar mensaje de error de inicio de sesión' , async ({ página }) => {

        espera la página . waitForSelector ( 'texto=Iniciar sesión' , { estado : 'visible' });

        espera la página . localizador ( 'texto=Iniciar sesión' ). primero (). haga clic ();

        espera la página . waitForSelector ( '#user\_email\_login' )

        espera la página . localizador ( '#user\_email\_login' ). escriba ( 'ejemplo1@ejemplo.com' );

        espera la página . localizador ( '#user\_password' ). escriba ( 'contraseña de ejemplo' );

        espera la página . localizador ( '#user\_submit' ). haga clic ();

        const errorMessage = await ( espera página . locator ( "//input[@id='user\_password']/../div[@class='error-msg']" ). textContent ()). recortar ();

        consola . log ( "Mensaje de error de inicio de sesión del navegador: " + mensaje de error );

        esperar ( mensaje de error ). toBe ( 'Contraseña no válida' );

    });

});

**Paso 7: ejecutar el script de prueba de dramaturgo**

Como se mencionó anteriormente, durante la instalación, Playwright crea **playwright.config.ts** ( *playwright.config.ts*es el archivo de configuración global) que tendrá algunas configuraciones. De forma predeterminada, la configuración global contiene el valor que se ejecutará en todos los navegadores. Se ejecuta en los tres navegadores diferentes cuando ejecuta la prueba Playwright. No lo necesitamos, por lo que debemos eliminar esa opción.

* Navegue hasta **playwright.config.ts**. Comenta la opción que comienza con **proyectos: [**

Si desea hacer lo anterior, copie y pegue la siguiente línea de código en **playwright.config.ts**

//dramaturgo.config.ts

tipo de importación { PlaywrightTestConfig } de '@playwright/test' ;

configuración constante : DramaturgoTestConfig = {

  testDir : './pruebas' ,

  tiempo de espera : 30 \* 1000 ,

  esperar : {

    tiempo de espera : 5000

  },

  reportero : 'html' ,

  utilizar : {

    tiempo de espera de acción : 0 ,

    rastreo : 'en el primer reintento' ,

  },

};

exportar configuración predeterminada ;

Entonces, ha configurado **playwright.config.ts**y estamos listos para ejecutar el primer script de prueba de Playwright.

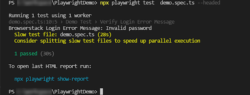
Ejecute Playwright Spec usando el siguiente comando

npx playwright test  demo.spec.ts –headed

Entendamos el comando anterior:

* Estamos especificando qué archivo de prueba necesitamos ejecutar, es decir **, *demo.spec.ts.***No mencione ningún nombre de archivo de especificaciones si desea ejecutar todos los archivos de especificaciones dentro de la carpeta de pruebas, no mencione ningún nombre de archivo de especificaciones.
* Playwright se ejecuta en modo sin cabeza de forma predeterminada, por lo que especificamos –headed ***para*** que se ejecute en modo con cabeza.
* No mencionamos ningún nombre de navegador; De forma predeterminada, la prueba Playwright se ejecuta en el navegador **Chromium.**

Una vez que ejecuta el comando anterior, las pruebas de Playwright comienzan a ejecutarse en el navegador Chromium.

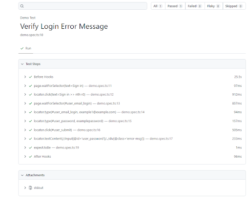


**Paso 8: ver el informe**

En nuestro **playwright.config.ts**, especificamos el informe HTML, por lo que después de ejecutar la prueba Playwright, el informe HTML se genera automáticamente en la carpeta **playwright-report**.

Para ver el informe HTML generado simplemente escriba:

npx playwright show-report



**Pruebas de dramaturgos en BrowserStack**

También se pueden ejecutar todas las pruebas [de dramaturgo](https://www.browserstack.com/docs/automate/playwright) en BrowserStack . Siga los pasos a continuación para ejecutar las pruebas de muestra en este repositorio:

1. Clona este repositorio usando git clone <https://github.com/browserstack/playwright-browserstack.git>
2. Vaya al directorio playwright-test usando ***cd playwright-test***
3. Instale las dependencias usando ***npm install***
4. Introduzca sus credenciales en el archivo **fixtures.js** en la parte de mayúsculas.
5. Para ejecutar sus pruebas en BrowserStack, debe asegurarse de haber configurado correctamente los proyectos en el archivo **playwright.config.js** .
6. Ejecute el script de prueba de muestra utilizando **npm test,** que ejecuta todas las pruebas dentro del directorio de pruebas en cinco navegadores en BrowserStack.
7. Si desea realizar pruebas en localhost u otros sitios web privados, ejecute el script de prueba de muestra usando **npm run test:local,** que ejecuta todas las pruebas dentro del directorio de pruebas en cinco navegadores en BrowserStack.
8. Para ejecutar sus pruebas localmente, debe configurar los proyectos sin el nombre **@browserstack en el archivo playwright.config.js** .

Para obtener detalles completos, visite la [página de Github](https://github.com/browserstack/playwright-browserstack) .

**Para concluir,**

Playwright es un sólido marco de automatización de pruebas que le permite realizar pruebas de end 2 end en los principales navegadores. Está equipado con múltiples funciones, como resiliencia, espera automática, captura de seguimiento de prueba, etc., compatibles con BrowserStack.

* La nube de dispositivos reales de BrowserStack proporciona más de 3000 navegadores y dispositivos reales para realizar pruebas instantáneas bajo demanda.
* La nube también proporciona [integraciones](https://www.browserstack.com/integrations) con herramientas CI/CD populares como Jira, Jenkins, TeamCity, Travis CI y más.
* [Además, las herramientas de depuración](https://www.browserstack.com/press/browserstack-launches-comprehensive-debugging-tools-and-expands-native-mobile-functionality) integradas permiten a los evaluadores identificar y resolver errores de inmediato.

Ejecute cientos de [pruebas de Playwright](https://www.browserstack.com/automate/playwright) en paralelo en cuestión de minutos. Céntrese en las pruebas, no en la infraestructura de pruebas, sin necesidad de configuraciones ni mantenimiento.

<https://www.browserstack.com/guide/playwright-tutorial>