Curso de Python y Selenium

Historia de selenium

Que es selenium?

Es una suite para automatizar procesos en el navegador.

Todo lo demas queda a lo que quieres aplicar.

Comienza en 2004, con Jason Huggins.

La suite de selenium

-Selenium IDE ⇒

sin escribir lineas de codigo solo dando clicks para implementar acciones. Imprime reportes. SOLO para Chrome y firefox. No soporta DataDriven Testing

-Selenium Remote Control ⇒

sin soporte solo para mantenimiento, Operaciones lógicas y condicionales. Soporta Data Driven Testing. Mas complejo de instalar. Todo se maneja a través de una API. Navegación muy ro-botica.

-Selenium Webdriver ⇒

conectarse a una api de alto nivel para controlar al navegador. Comunicación directa con el servidor. Interacción muy realista podemos actuar como lo haría un usuario. No soporte para navegadoras rápido. No genera reportes por si solo. Necesita programar.

-Selenium Grid ⇒

Se utiliza junto con Selenium RC para trabajar con otras circunstancias en paralelo. Prueba en distintos entornos.

Ventajas

- Compatible con gran cantidad de navegadores web
- Compatible con Java, C#, Kotlin, Perl, Php, python, ruby, Javascript

Desventajas

- No compatible con golang
- No herramienta especifica de testing, ni web scraping

Otras herramientas.

Puppeteer ⇒ Herramienta mas especializada con google. No archivos externos. Solo javascript y solo funciona con chrome.

Cypreessio ⇒ Documentación muy buena. Muy ágil en pruebas End to End. Excelente manejo de asincronismo.

Requisitos:

- Python 3.6 minimo
- Selenium ⇒ pip3 install selenium
- PyUnitReport ⇒ librería para generar reportes en HTML

Unittest libreria de python

Test Fixture ⇒ Prepara antes y después de la automatización. Prepara el entorno y realiza algo después de el caso de prueba.

SetUp ⇒ método

los métodos de prueba deben iniciar con test_descripción

TearDown ⇒ métodos

Test Suite ⇒ Una colección de distintos pruebas o automatizaciones en un solo archivo ejecutando de forma secuencial

Test runner ⇒ orquestador de la ejecución

Test Report ⇒ Resumen de resultados.

---

implicity_wait() ⇒ tiempo de espera hasta de cierto tiempo antes de mandar "No Such Element Exception" es decir no encontró el elemento.

select.options ⇒ cantidad de opciones que tiene un dropdow

is_displayed() ⇒ esta visible

is_enabled() ⇒ esta activo

decorador ⇒ @classmethod ⇒ cambiando self por cls

driver.quit() ⇒ cierra la prueba

Distintos selectores para poder interactuar con ellos:

Recomendaciones:

x-path ⇒ hace que nuestro texto sea muy largo y poco prolijo por otro lado los sitios dinámicos suelen cambiar y suele hacer que por lo mismo el x-path también cambie.

Asertions:

Métodos que permiten validar un valor esperado dentro del test. En caso de que se cumpla el test continua de lo contrario falla.

Test Suite:

Son pruebas unificadas en una sola prueba o archivo para que se corran de forma secuencial.

Clase WebElment:

Permite interactuar específicamente con elementos de los sitios web como textbox, text area, button, radio button, checkbox etc...

Clase WebDriver:

Interacción directa con la ventana del navegador y sus elementos relacionados, como pop-ups o alerts.

Manejar Alert y Pop-Up

alert.accept() ⇒ acepta el alert

alert.send_keys() ⇒ simula escribir o presionar teclas e un elemento

aler.dismiss() ⇒ Rechaza el mensaje enviado por el alert haciendo click en cancelar.

Navegación

back()

foward()

refresh()

Demoras:

- Implícitas ⇒ busca uno o varios elementos en el DOM si no se encuentran disponibles por la cantidad de tiempo asignado.

- Explicitas ⇒ busca que una condición determinada se cumpla y después continua con la prueba.
-

By

Ayuda a hacer referencia a través de su selectores e interactuar.

Recomendación:

Has uso de los try except en sitios donde el contenido sea dinámico y pueda haber algunos fallos debido a que el contenido no siempre es el mismo.

Recuerda que los inputs introducidos por el usuario son posibles errores, debido al mal uso de ellos.

Patrones de diseno de Testing

Data Driven Testing

TDD ⇒ basado en pruebas que se hicieron previamente

DDT ⇒ basado en software que ya esta escrito para hacer validaciones. Data Driven testing ⇒ Colocar datos para soportar multiples pruebas.

Page Object Model (POM)

patron de diseno de testing

como funciona ⇒ manejaremos archivos independientes llamados pages.

Beneficios

código re-utilizable ⇒ un solo archivos

las pruebas son mas flexibles, legibles y vigentes