Enlaces de Selenio Python

Lanzamiento 2

Baiju Muthukadan

05 de abril de 2024

CONTENIDO

Autor

Baiju Muthukadan

Nota: Esta no es una documentación oficial. Si desea contribuir a esta documentación, puedebifurca este proyecto en GitHub y envía solicitudes de extracción. También puede enviar sus comentarios a mi correo electrónico: baiju.m.mail AT gmail DOT com. Hasta ahora, más de 60 miembros de la comunidad han contribuido a este proyecto (consulte las solicitudes de extracción cerradas). ¡Animo a los contribuyentes a agregar más secciones y convertirla en una documentación increíble! Si conoce alguna traducción de este documento, envíe un PR para actualizar la lista a continuación.

Traducciones:

- Chino
- japonés

CAPÍTULO

UNO

INSTALACIÓN

1.1 Introducción

Los enlaces de Selenium Python proporcionan una API simple para escribir pruebas funcionales/de aceptación utilizando Selenium WebDriver. A través de Selenium Python API puedes acceder a todas las funcionalidades de Selenium WebDriver de forma intuitiva.

Los enlaces de Selenium Python proporcionan una API conveniente para acceder a Selenium WebDrivers como Firefox, Ie, Chrome, Remote, etc. Las versiones actuales de Python compatibles son 3.5 y superiores.

Esta documentación explica la API de Selenium 2 WebDriver. La API de Selenium 1/Selenium RC no se trata aquí.

1.2 Instalación de enlaces de Python para Selenium

Usarpepitapara instalar el paquete de selenio. Python 3 tiene pip disponible en elbiblioteca estándar. Usando*pepita*, puedes instalar selenio de esta manera:

```
pip instalar selenio
```

Puedes considerar usarentorno virtualpara crear entornos Python aislados. Python 3 tienevenvque es casi lo mismo que virtualenv.

También puede descargar enlaces de Python para Selenium desdePágina de PyPI para el paquete de selenio. e instalar manualmente.

1.3 Instrucciones para usuarios de Windows

1. Instale Python 3 usando elMSI disponible en la página de descarga de python.org.
2. Inicie un símbolo del sistema usando elcmd.exeprograma y ejecuta elpepitacomando como se indica a continuación para instalar*selenio*.

```
C:\Python39\Scripts\pip.exe instalar selenio
```

Ahora puedes ejecutar tus scripts de prueba usando Python. Por ejemplo, si creó un script basado en Selenium y lo guardó dentroC:\my_selenium_script.py,puedes ejecutarlo así:

```
C:\Python39\python.exe C:\mi_selenium_script.py
```

1.4 Instalación desde fuentes de Git

Para construir Selenium Python a partir del código fuente, clonarel repositorio oficial. Contiene el código fuente de todas las versiones oficiales de Selenium, como Python, Java, Ruby y otros. El código Python reside en /pydirectorio. Para construir, también necesitarás elBazelsistema de construcción.

Nota: Actualmente, a medida que Selenium se acerca a la versión 4.0.0, requiere Bazel 3.2.0 (Instrucciones de instalación), aunque la versión 3.3.0 ya está disponible.

Para crear una rueda a partir de las fuentes, ejecute el siguiente comando desde la raíz del repositorio:

```
bazel//py:selenio-rueda
```

Este comando preparará el código fuente con algunos archivos JS preprocesados que necesitan algunos módulos del controlador web y compilará el archivo .whlpaquete dentro del ./bazel-bin/py/directorio. Después puedes utilizarpepitapara instalarlo.

1.5 Conductores

Selenium requiere un controlador para interactuar con el navegador elegido. Firefox, por ejemplo, requieregeckodriver, que debe instalarse antes de poder ejecutar los siguientes ejemplos. Asegúrate de que esté en tu*CAMINO*, por ejemplo, colóquelo en*/usuario/bin*o */usr/local/bin*.

No seguir este paso te dará un error*selenium.common.exceptions.WebDriverException: Mensaje: el ejecutable 'geckodriver' debe estar en PATH.*

Otros navegadores compatibles tendrán sus propios controladores disponibles. A continuación encontrará enlaces a algunos de los controladores de navegador más populares.

Cromo:	https://sites.google.com/chromium.org/driver/
Borde:	https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
Firefox:	https://github.com/mozilla/geckodriver/releases
Safari:	https://webkit.org/blog/6900/webdriver-support-in-safari-10/

Para obtener más información sobre la instalación del controlador, consulte ladocumentación oficial.

A partir de la versión4.6.0 (4 de noviembre de 2022) el selenio viene con**Gerente de selenio**Embalado en distribución.

Gerente de selenioes una nueva herramienta que ayuda a hacer funcionar un entorno de trabajo**Selenio**fuera de la caja:

- descubre, descarga y almacena en caché automáticamenteconductoresrequerido por Selenium cuando estosconductoresno están disponibles;
- descubre, descarga y almacena en caché automáticamentenavegadoresimpulsado con Selenium (Chrome, Firefox y Edge) cuando estosnavegadoresno están instalados en el sistema local.

Por ejemplo, para ver el resultado de**Gerente de selenio**funciona, simplemente ejecuta cualquier script de Selenium sin configuración previa del controlador y explora*~/.caché/selenio*.

Más sobre**Gerente de selenio**puedes leer en eldocumentaciónyblog.

1.6 Descarga del servidor Selenium

Nota: El servidor Selenium solo es necesario si desea utilizar el WebDriver remoto. Ver el*Usando Selenium con WebDriver remoto*sección para más detalles. Si es un principiante en el aprendizaje de Selenium, puede omitir esta sección y continuar con el siguiente capítulo.

El servidor Selenium es un programa Java. Se recomienda Java Runtime Environment (JRE) 1.6 o una versión más reciente para ejecutar el servidor Selenium.

Puede descargar el servidor Selenium 2.x desdepágina de descarga del sitio web de selenio. El nombre del archivo debería ser algo como esto:selenium-servidor-independiente-2.xxjar.Siempre puedes descargar la última versión 2.x del servidor Selenium.

Si Java Runtime Environment (JRE) no está instalado en su sistema, puede descargar elJRE del sitio web de Oracle. Si está utilizando un sistema GNU/Linux y tiene acceso de root en su sistema, también puede utilizar las instrucciones de su sistema operativo para instalar JRE.

Si*java*El comando está disponible en la RUTA (variable de entorno), puede iniciar el servidor Selenium usando este comando:

```
Java -tarro de selenio-servidor-autónomo-2.incógnita.incógnita.frasco
```

Reemplazar*2.xx*con la versión real del servidor Selenium que descargaste del sitio.

Si JRE está instalado como usuario no root y/o si no está disponible en la RUTA (variable de entorno), puede escribir la ruta relativa o absoluta al*java*dominio. De manera similar, puede proporcionar una ruta relativa o absoluta al archivo jar del servidor Selenium. Entonces, el comando se verá así:

```
/camino/a/Java-frasco/camino/a/selenio-servidor-autónomo-2.incógnita.incógnita.frasco
```

CAPÍTULO

DOS

EMPEZANDO

2.1 Uso sencillo

Si ha instalado enlaces de Selenium Python, puede comenzar a usarlo desde Python de esta manera.

```
deselenioimportarcontrolador web
deselenium.webdriver.common.keysimportarLlaves de
selenio.webdriver.common.byimportarPor

conductor=controlador web.Controlador Firefox().
conseguir("http://www.python.org") afirmar"Pitón"en
conductor.elemento de título=conductor.
encontrar_elemento(Por.NOMBRE,"q") elemento.claro()

elemento.enviar_claves("pycon") elemento.
enviar_claves(Claves.DEVOLVER)
afirmar"No se encontraron resultados."no enconductor.controlador de
fuente de página.cerca()
```

El script anterior se puede guardar en un archivo (por ejemplo: *-python_org_search.py*), entonces se puede ejecutar así:

```
python python_org_búsqueda.py
```

El*pitón*que estás ejecutando debe tener el*selenio*módulo instalado.

2.2 Ejemplo explicado

El*selenio.webdriver*El módulo proporciona todas las implementaciones de WebDriver. Las implementaciones de WebDriver actualmente admitidas son Firefox, Chrome, IE y Remote. El*Llaves*La clase proporciona teclas en el teclado como RETURN, F1, ALT, etc.*Por*La clase se utiliza para localizar elementos dentro de un documento.

```
deselenioimportarcontrolador web
deselenium.webdriver.common.keysimportarLlaves de
selenio.webdriver.common.byimportarPor
```

A continuación, se crea la instancia de Firefox WebDriver.

```
conductor=controlador web.Firefox()
```

El*conductor.obtener*El método navegará a una página proporcionada por la URL. WebDriver esperará hasta que la página se haya cargado por completo (es decir, que se haya activado el evento "onload") antes de devolver el control a su prueba o secuencia de comandos.*Tenga en cuenta que si su página utiliza mucho AJAX durante la carga, es posible que WebDriver no sepa cuándo se ha cargado por completo.*:

```
conductor.conseguir("http://www.python.org")
```

La siguiente línea es una afirmación para confirmar que el título tiene la palabra "Python":

```
afirmar"Pitón"enconductor.título
```

WebDriver ofrece varias formas de buscar elementos utilizando el*encontrar_elemento*método. Por ejemplo, el elemento de texto de entrada se puede ubicar por su*nombre*atributo usando el*encontrar_elemento*método y utilizando By.NAME como su primer parámetro. Una explicación detallada de cómo encontrar elementos está disponible en el*Localización de elementos*capítulo:

```
elemento=conductor.encontrar_elemento(Por.NOMBRE,"q")
```

A continuación, enviamos claves, esto es similar a ingresar claves usando el teclado. Se pueden enviar claves especiales utilizando el*Llaves* clase importada de*selenium.webdriver.common.keys*. Para estar seguros, primero borraremos cualquier texto precargado en el campo de entrada (por ejemplo, "Buscar") para que no afecte nuestros resultados de búsqueda:

```
elemento.claro()
elemento.enviar_claves("pycon") elemento.
enviar_claves(Claves.DEVOLVER)
```

Después de enviar la página, debería obtener el resultado, si lo hay. Para asegurarse de que se encuentren algunos resultados, haga una afirmación:

```
afirmar"No se encontraron resultados."no enconductor.fuente_página
```

Finalmente, se cierra la ventana del navegador. También puedes llamar al*abandonar*método en lugar de*cerca*. El*abandonar*El método saldrá del navegador mientras que*cerca*cerrará una pestaña, pero si solo una pestaña estaba abierta, de forma predeterminada la mayoría de los navegadores se cerrarán por completo.:

```
conductor.cerca()
```

2.3 Usar Selenium para escribir pruebas

El selenio se utiliza principalmente para escribir casos de prueba. El*selenio*El paquete en sí no proporciona una herramienta/marco de prueba. Puede escribir casos de prueba utilizando el módulo unittest de Python. Alternativamente, puede considerarpytestpara exámenes de redacción.

En este capítulo utilizamos*prueba unitaria*como marco de elección. Aquí está el ejemplo modificado que utiliza el módulo unittest. Esta es una prueba para el*python.org*funcionalidad de búsqueda:

```
importarprueba unitaria
deselenioimportarcontrolador web
deselenium.webdriver.common.keysimportarLlaves de
selenio.webdriver.common.byimportarPor

clasePythonOrgSearch(prueba unitaria.Caso de prueba):

    definiciónconfiguración(ser):
        ser.conductor=controlador web.Firefox()

    definióntest_search_in_python_org(ser):
```

(continúa en la página siguiente)

(continúa de la página anterior)

```
        conductor=ser.conductor
        conductor.conseguir("http://www.python.org") ser.afirmar
        en ("Pitón", conductor.título) elemento=conductor.
        encontrar_elemento(Por.NOMBRE,"q") elemento.
        enviar_claves("pycon") elemento.enviar_claves(Claves.
        DEVOLVER)
        ser.afirmarNotIn("No se encontraron resultados"., conductor.fuente_página)


    definicióndemoler(ser):
        ser.conductor.cerca()

si__nombre__=="__principal__":
    prueba unitaria.principal()
```

Puede ejecutar el caso de prueba anterior desde un shell como este:

```
python test_python_org_search.py
.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Corrió1pruebaen15.566s

DE ACUERDO
```

El resultado anterior muestra que la prueba se completó con éxito.

Nota: Para ejecutar la prueba anterior en IPython o Jupyter, debe pasar un par de argumentos al*principal*funcionar como se muestra a continuación:

```
prueba unitaria.principal(argv)=['primer-argumento-es-ignorado'], salida=FALSO)
```

2.4 Tutorial del ejemplo

Inicialmente, se importan todos los módulos básicos necesarios. Elprueba unitariaEl módulo es un módulo Python integrado basado en JUnit de Java. Este módulo proporciona el marco para organizar los casos de prueba. El*selenio.webdriver*El módulo proporciona todas las implementaciones de WebDriver. Las implementaciones de WebDriver admitidas actualmente son: Firefox, Chrome, IE y Remote. El*Llaves*La clase proporciona teclas en el teclado como RETURN, F1, ALT, etc.*Por*La clase se utiliza para localizar elementos dentro de un documento.

```
importarprueba unitaria
deselenioimportarcontrolador web
deselenium.webdriver.common.keysimportarLlaves de
selenio.webdriver.common.byimportarPor
```

La clase de caso de prueba se hereda de*prueba unitaria.Case de prueba*. Heredando de la*Caso de prueba*la clase es la manera de decir*prueba unitaria* módulo que este es un caso de prueba:

```
clasePythonOrgSearch(prueba unitaria.Caso de prueba):
```

El*configuración*El método es parte de la inicialización. Este método será llamado antes de cada función de prueba que vaya a escribir en esta clase de caso de prueba. Aquí está creando una instancia de Firefox WebDriver.

```
definiciónconfiguración(ser):
    ser.conductor=controlador web.Firefox()
```

Este es el método del caso de prueba. El método del caso de prueba siempre debe comenzar con caracteres.*prueba*. La primera línea dentro de este método crea una referencia local al objeto controlador creado en*configuración*método.

```
definiciónprueba_buscar_en_python_org(ser):
    conductor=ser.conductor
```

El*conductor.obtener*El método navegará a una página proporcionada por la URL. WebDriver esperará hasta que la página se haya cargado por completo (es decir, que se haya activado el evento "onload") antes de devolver el control a su prueba o secuencia de comandos.*Tenga en cuenta que si su página utiliza mucho AJAX durante la carga, es posible que WebDriver no sepa cuándo se ha cargado por completo.*:

```
conductor.conseguir("http://www.python.org")
```

La siguiente línea es una afirmación para confirmar que el título tiene la palabra "Python":

```
ser.afirmar en ("Pitón", conductor.título)
```

WebDriver ofrece varias formas de buscar elementos utilizando el*encontrar_elemento*método. Por ejemplo, el elemento de texto de entrada se puede ubicar por su*nombre*atributo usando el*encontrar_elemento*método. La explicación detallada de los elementos encontrados está disponible en el*Localización de elementos*capítulo:

```
elemento=conductor.encontrar_elemento(Por.NOMBRE,"q")
```

A continuación, enviamos claves, esto es similar a ingresar claves usando el teclado. Se pueden enviar claves especiales utilizando el*Llaves* clase importada de*selenium.webdriver.common.keys*:

```
elemento.enviar_claves("pycon") elemento.
enviar_claves(Claves.DEVOLVER)
```

Después de enviar la página, debería obtener el resultado según la búsqueda, si la hay. Para asegurarse de que se encuentren algunos resultados, haga una afirmación:

```
ser.afirmarNotIn("No se encontraron resultados"., conductor.fuente_página)
```

El*demoler*El método será llamado después de cada método de prueba. Este es un lugar para realizar todas las acciones de limpieza. En el método actual, la ventana del navegador está cerrada. También puedes llamar al*abandonar*método en lugar de*cerca*. El*abandonar*El método saldrá de todo el navegador, mientras que*cerca* cerrará una pestaña, pero si es la única pestaña abierta, de forma predeterminada la mayoría de los navegadores se cerrarán por completo.:

```
definicióndemoler(ser):
    ser.conductor.cerca()
```

Las líneas finales son un código repetitivo para ejecutar el conjunto de pruebas:

```
si__nombre__=="__principal__":
    prueba unitaria.principal()
```

2.5 Usando Selenium con WebDriver remoto

Para utilizar el WebDriver remoto, debe tener el servidor Selenium ejecutándose. Para ejecutar el servidor, use este comando:

```
Java-tarro de selenio-servidor-autónomo-2.incógnita.incógnita.frasco
```

Mientras ejecuta el servidor Selenium, puede ver un mensaje similar a este:

```
15:43:07.541INFORMACIÓN-Las instancias de RemoteWebDriver deben conectarse a: http://127.0.0.1:4444/
  →wd/centro
```

La línea anterior dice que puede utilizar esta URL para conectarse al WebDriver remoto. A continuación se muestran algunos ejemplos:

```
deselenioimportarcontrolador web

conductor=controlador web.Remoto(
    comando_ejecutor='http://127.0.0.1:4444/wd/hub', opciones=
    controlador web.Opciones de Chrome()
)

conductor=controlador web.Remoto(
    comando_ejecutor='http://127.0.0.1:4444/wd/hub', opciones=
    controlador web.Opciones de Firefox()
)
```

CAPÍTULO

TRES

NAVEGANDO

Lo primero que querrá hacer con WebDriver es navegar hasta un enlace. La forma normal de hacerlo es llamandoconseguir método:

```
conductor.conseguir("http://www.google.com")
```

WebDriver esperará hasta que la página se haya cargado por completo (es decir, elcargarevento se ha disparado) antes de devolver el control a su prueba o secuencia de comandos.*Tenga en cuenta que si su página utiliza mucho AJAX durante la carga, es posible que WebDriver no sepa cuándo se ha cargado por completo.*. Si necesita asegurarse de que dichas páginas estén completamente cargadas, puede utilizar*murga*.

3.1 Interactuando con la página

El simple hecho de poder ir a lugares no es muy útil. Lo que realmente nos gustaría hacer es interactuar con las páginas o, más específicamente, con los elementos HTML dentro de una página. En primer lugar, necesitamos encontrar uno. WebDriver ofrece varias formas de encontrar elementos. Por ejemplo, dado un elemento definido como:

```
<tipo de entrada="texto"nombre="contraseña"identificación="contraseña-id"/>
```

puedes encontrarlo usando cualquiera de:

```
elemento=conductor.encontrar_elemento(Por.IDENTIFICACIÓN,"contraseña-
id") elemento=conductor.encontrar_elemento(Por.NOMBRE,"contraseña")
elemento=conductor.encontrar_elemento(Por.XPATH,"//entrada[@id='contraseña-id']")
elemento=conductor.encontrar_elemento(Por.CSS_SELECTOR,"entrada#contraseña-id")
```

También puedes buscar un enlace por su texto, ¡pero cuidado! ¡El texto debe coincidir exactamente! También debes tener cuidado al usar*XPATH en WebDriver*. Si hay más de un elemento que coincide con la consulta, solo se devolverá el primero. Si no se puede encontrar nada, unNinguna excepción de elemento talserá levantado.

WebDriver tiene una API "basada en objetos"; Representamos todo tipo de elementos utilizando una misma interfaz. Esto significa que, aunque es posible que vea muchos métodos posibles que podría invocar cuando presione la combinación de teclas de autocompletar de su IDE, no todos tendrán sentido o serán válidos. ¡No te preocupes! WebDriver intentará hacer lo correcto y, si llama a un método que no tiene sentido ("setSelected()" en una etiqueta "meta", por ejemplo), se generará una excepción.

Entonces, tienes un elemento. ¿Qué puedes hacer con él? En primer lugar, es posible que desees ingresar algo de texto en un campo de texto:

```
elemento.enviar_claves("algo de texto")
```

Puede simular presionar las teclas de flecha usando la clase "Teclas":

```
elemento.enviar_claves("y algunos", Llaves.FLECHA_ABAJO)
```

Es posible llamar *enviar_claves* en cualquier elemento, lo que permite probar atajos de teclado como los utilizados en GMail. Un efecto secundario de esto es que escribir algo en un campo de texto no lo borrará automáticamente. En cambio, lo que escriba se agregará a lo que ya está allí. Puede borrar fácilmente el contenido de un campo de texto o área de texto con el *claro* método:

```
elemento.claro()
```

3.2 Rellenar formularios

Ya hemos visto cómo ingresar texto en un área de texto o campo de texto, pero ¿qué pasa con los otros elementos? Puede "alternar" el estado del menú desplegable y puede usar "setSelected" para configurar algo como un *OPCIÓN* etiqueta seleccionada. Tratando con *SELECCIONAR* Las etiquetas no están tan mal:

```
elemento=conductor.encontrar_elemento(Por.XPATH,"//seleccionar[@nombre='nombre']")
todas_opciones=elemento.buscar_elementos(Por.ETIQUETA_NOMBRE,"opción") para
opciónentodas_opciones:
    imprimir("El valor es: %s"%opción.obtener_atributo("valor"))
    opción.hacer clic()
```

Esto encontrará el primer elemento "SELECCIONAR" en la página y recorrerá cada una de sus OPCIONES, imprimiendo sus valores y seleccionando cada una de ellas.

Como puede ver, esta no es la forma más eficaz de tratar con elementos SELECT. Las clases de soporte de WebDriver incluyen una llamada "Select", que proporciona métodos útiles para interactuar con estos:

```
deselenio.webdriver.support.uiimportarseleccionar seleccionar=
Seleccionar (conductor.encontrar_elemento(Por.NOMBRE,'nombre'))
seleccionar.select_by_index(índice) seleccionar.
seleccionar_por_texto_visible("texto") seleccionar.
seleccionar_por_valor(valor)
```

WebDriver también proporciona funciones para deseleccionar todas las opciones seleccionadas:

```
seleccionar=Seleccionar (conductor.encontrar_elemento(Por.IDENTIFICACIÓN,
'identificación')) seleccionar.deseleccionar_todo()
```

Esto anulará la selección de todas las OPCIONES de ese SELECT en particular en la página.

Supongamos que en una prueba necesitamos la lista de todas las opciones seleccionadas predeterminadas. La clase Select proporciona un método de propiedad que devuelve una lista:

```
seleccionar=Seleccionar (conductor.encontrar_elemento(Por.XPATH,"//seleccionar[@nombre='nombre']")) todas las
opciones_seleccionadas=seleccionar.todas las opciones_seleccionadas
```

Para obtener todas las opciones disponibles:

```
opciones=seleccionar.opciones
```

Una vez que haya terminado de completar el formulario, probablemente desee enviarlo. Una forma de hacerlo sería buscar el botón "enviar" y hacer clic en él:

```
# Supongamos que el botón tiene el ID "enviar" :) conductor.
encontrar_elemento(Por.IDENTIFICACIÓN,"entregar").hacer clic()
```

Alternativamente, WebDriver tiene el método conveniente "enviar" en cada elemento. Si llama a esto en un elemento dentro de un formulario, WebDriver recorrerá el DOM hasta encontrar el formulario adjunto y luego llamará a enviar. Si el elemento no está en una forma, entonces elNinguna excepción de elemento talse levantará:

```
elemento.entregar()
```

3.3 Arrastrar y soltar

Puede usar arrastrar y soltar, ya sea moviendo un elemento una cierta cantidad o hacia otro elemento:

```
elemento=conductor.encontrar_elemento(Por.NOMBRE,"fuente")
objetivo=conductor.encontrar_elemento(Por.NOMBRE,"objetivo")

deselenio.webdriverimportarCadenas de acción
cadenas_de_acción=Cadenas de acción (controlador)
cadenas_de_accion.arrastrar y soltar (elemento, destino).llevar a cabo()
```

3.4 Moverse entre ventanas y marcos

Es raro que una aplicación web moderna no tenga marcos o esté restringida a una sola ventana. WebDriver admite moverse entre ventanas con nombre utilizando el método "switch_to.window":

```
conductor.cambiar_a.ventana("nombre de la ventana")
```

todas las llamadas aconductorahora se interpretará como dirigido a la ventana en particular. ¿Pero cómo sabes el nombre de la ventana? Echa un vistazo al javascript o enlace que lo abrió:

```
<un href="en algún lugar.html"objetivo="nombre de la ventana">Haga clic aquí paraabiertouna nueva ventana</a>
```

Alternativamente, puede pasar un "identificador de ventana" al método "switch_to.window()". Sabiendo esto, es posible iterar sobre cada ventana abierta de esta manera:

```
paramanejarenconductor.manijas_ventana:
    conductor.cambiar_a.ventana (manija)
```

También puedes pasar de un cuadro a otro (o a iframes):

```
conductor.cambiar_a.marco("nombre del marco")
```

Es posible acceder a subcuadros separando la ruta con un punto y también puede especificar el marco por su índice. Eso es:

```
conductor.cambiar_a.marco("nombreDeMarco.0.niño")
```

iría al marco llamado "secundario" del primer subtrama del marco llamado "nombre de marco".**Todos los fotogramas se evalúan como desde *arriba*.**

Una vez que hayamos terminado de trabajar en los marcos, tendremos que volver al marco principal, lo cual se puede hacer usando:

```
conductor.cambiar_a.contenido_predeterminado()
```

3.5 Cuadros de diálogo emergentes

Selenium WebDriver tiene soporte integrado para manejar cuadros de diálogo emergentes. Después de haber activado una acción que abriría una ventana emergente, puede acceder a la alerta con lo siguiente:

```
alerta=conductor.cambiar_a.alerta
```

Esto devolverá el objeto de alerta actualmente abierto. Con este objeto, ahora puede aceptar, descartar, leer su contenido o incluso escribir en un mensaje. Esta interfaz funciona igualmente bien en alertas, confirmaciones y avisos. Consulte la documentación de la API para obtener más información.

3.6 Navegación: historia y ubicación

Anteriormente, cubrimos la navegación a una página usando el comando "obtener" (controlador.get("http://www.ejemplo.com")). Como ha visto, WebDriver tiene varias interfaces más pequeñas centradas en tareas, y la navegación es una tarea útil. Para navegar a una página, puede utilizar*conseguir*método:

```
conductor.conseguir("http://www.ejemplo.com")
```

Para avanzar y retroceder en el historial de su navegador:

```
conductor.adelante()
conductor.atrás()
```

Tenga en cuenta que esta funcionalidad depende completamente del controlador subyacente. Es posible que suceda algo inesperado al llamar a estos métodos si está acostumbrado al comportamiento de un navegador sobre otro.

3.7 Galletas

Antes de pasar a la siguiente sección del tutorial, es posible que le interese comprender cómo utilizar las cookies. En primer lugar, debe estar en el dominio para el que la cookie será válida:

```
# Ir al dominio correcto conductor.
conseguir("http://www.ejemplo.com")

# Ahora configura la cookie. Este es válido para todo el dominio.
galleta={'nombre':'foo','valor':'bar'} conductor.agregar_cookie(cookie)


# Y ahora genera todas las cookies disponibles para la URL actual conductor.
obtener_cookies()
```

CAPÍTULO CUATRO

ELEMENTOS DE LOCALIZACIÓN

Existen varias estrategias para localizar elementos en una página. Puedes utilizar el más apropiado para tu caso. Selenium proporciona el siguiente método para localizar elementos en una página:

- *encontrar_elemento*

Para buscar varios elementos (estos métodos devolverán una lista):

- *encontrar_elementos*

Uso de ejemplo:

```
deselenio.webdriver.common.byimportarPor

conductor.encontrar_elemento(Por.XPATH,'//botón[text()="Algo de texto"]')
conductor.buscar_elementos(Por.XPATH,'//botón')
```

Los atributos disponibles para el*Por*Las clases se utilizan para localizar elementos en una página. Estos son los atributos disponibles para*Por* clase:

```
IDENTIFICACIÓN="identificación"
NOMBRE="nombre"
XPATH="xpath"
ENLACE_TEXT="texto de enlace" PARTIAL_LINK_TEXT=
"texto de enlace parcial" ETIQUETA_NOMBRE="nombre de
etiqueta" CLASE_NOMBRE="nombre de clase"
CSS_SELECTOR="selector css"
```

La clase 'Por' se utiliza para especificar qué atributo se utiliza para localizar elementos en una página. Estas son las diversas formas en que se utilizan los atributos para ubicar elementos en una página:

```
encontrar_elemento(Por.IDENTIFICACIÓN,"identificación") encontrar_elemento(Por.
NOMBRE,"nombre") encontrar_elemento(Por.XPATH,"xpath") encontrar_elemento(Por.
ENLACE_TEXTO,"texto de enlace") encontrar_elemento(Por.PARTIAL_LINK_TEXT,"texto
de enlace parcial") encontrar_elemento(Por.ETIQUETA_NOMBRE,"nombre de etiqueta")
encontrar_elemento(Por.CLASE_NOMBRE,"nombre de clase") encontrar_elemento(Por.
CSS_SELECTOR,"selector css")
```

Si desea localizar varios elementos con el mismo atributo reemplace find_element con find_elements.

4.1 Localización por ID

Utilízalo cuando conozcas el*identificación*atributo de un elemento. Con esta estrategia, el primer elemento con una coincidencia*identificación*Se devolverá el atributo. Si ningún elemento tiene una coincidencia*identificación*atributo, unNinguna excepción de elemento talserá levantado.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <formaidentificación="formulario de inicio de sesión">
      <aportenombre="nombre de usuario"tipo="texto"/> <
      aportenombre="contraseña"tipo="contraseña"/>
      <aportenombre="continuar"tipo="entregar"valor="Acceso"/> </
    forma>
  </cuerpo>
</HTML>
```

El elemento de formulario se puede ubicar así:

```
formulario de inicio de sesión=conductor.encontrar_elemento(Por.IDENTIFICACIÓN,'formulario de inicio de sesión')
```

4.2 Localización por nombre

Utilízalo cuando conozcas el*nombre*atributo de un elemento. Con esta estrategia, el primer elemento con una coincidencia*nombre* Se devolverá el atributo. Si ningún elemento tiene una coincidencia*nombre*atributo, unNinguna excepción de elemento talserá levantado.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <formaidentificación="formulario de inicio de sesión">
      <aportenombre="nombre de usuario"tipo="texto"/> <
      aportenombre="contraseña"tipo="contraseña"/>
      <aportenombre="continuar"tipo="entregar"valor="Acceso"/>
      <aportenombre="continuar"tipo="botón"valor="Claro"/> </
    forma>
</cuerpo>
</HTML>
```

Los elementos de nombre de usuario y contraseña se pueden ubicar así:

```
nombre de usuario=conductor.encontrar_elemento(Por.NOMBRE,'nombre de
usuario') contraseña=conductor.encontrar_elemento(Por.NOMBRE,'contraseña')
```

Esto mostrará el botón "Iniciar sesión" tal como aparece antes del botón "Borrar":

```
botón_continuar=conductor.encontrar_elemento(Por.NOMBRE,'continuar')
```

4.3 Localización por XPath

XPath es el lenguaje utilizado para localizar nodos en un documento XML. Como HTML puede ser una implementación de XML (XHTML), los usuarios de Selenium pueden aprovechar este poderoso lenguaje para apuntar a elementos en sus aplicaciones web. XPath admite métodos simples de localización por atributos de identificación o nombre y los amplía abriendo todo tipo de nuevas posibilidades, como localizar la tercera casilla de verificación en la página.

Una de las razones principales para usar XPath es cuando no tiene un atributo de identificación o nombre adecuado para el elemento que desea ubicar. Puede utilizar XPath para ubicar el elemento en términos absolutos (no recomendado) o en relación con un elemento que tenga un atributo de identificación o nombre. Los localizadores XPath también se pueden utilizar para especificar elementos mediante atributos distintos de id y nombre.

Los XPath absolutos contienen la ubicación de todos los elementos desde la raíz (html) y, como resultado, es probable que fallen con solo el más mínimo ajuste de la aplicación. Al encontrar un elemento cercano con un atributo de identificación o nombre (idealmente un elemento principal), puede ubicar su elemento de destino según la relación. Es mucho menos probable que esto cambie y puede hacer que sus pruebas sean más sólidas.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <formaidentificación="formulario de inicio de sesión">
      <aportenombre="nombre de usuario"tipo="texto"/> <
      aportenombre="contraseña"tipo="contraseña"/>
      <aportenombre="continuar"tipo="entregar"valor="Acceso"/>
      <aportenombre="continuar"tipo="botón"valor="Claro"/> </
    forma>
</cuerpo>
</HTML>
```

Los elementos del formulario se pueden ubicar así:

```
formulario de inicio de sesión=conductor.encontrar_elemento(Por.XPATH,"/html/cuerpo/formulario[1]") formulario de
inicio de sesión=conductor.encontrar_elemento(Por.XPATH,"//formulario[1]") formulario de inicio de sesión=conductor
.encontrar_elemento(Por.XPATH,"//formulario[@id='loginForm']")
```

1. Ruta absoluta (se rompería si el HTML se cambiara sólo ligeramente)

2. Primer elemento de formulario en HTML.

3. El elemento de formulario con atributo.*identificación*empezar a*formulario de inicio de sesión*

El elemento de nombre de usuario se puede ubicar así:

```
nombre de usuario=conductor.encontrar_elemento(Por.XPATH,"//formulario[entrada/@nombre='nombre de usuario']")
nombre de usuario=conductor.encontrar_elemento(Por.XPATH,"//formulario[@id='loginForm']/entrada[1]") nombre de
usuario=conductor.encontrar_elemento(Por.XPATH,"//entrada[@nombre='nombre de usuario']")
```

1. Primer elemento de formulario con un elemento secundario de entrada con*nombre*empezar a*nombre de usuario*

2. Primer elemento secundario de entrada del elemento de formulario con atributo*identificación*empezar a*formulario de inicio de sesión*

3. Primer elemento de entrada con atributo.*nombre*empezar a*nombre de usuario*

El elemento del botón "Borrar" se puede ubicar así:

```
boton_claro=conductor.encontrar_elemento(Por.XPATH,"//entrada[@nombre='continuar'][@tipo='botón']")
botón_claro=conductor.encontrar_elemento(Por.XPATH,"//formulario[@id='loginForm']/entrada[4]")
```

1. Entrada con atributo *nombre* empezar a *continua* y atributo *tipo* empezar a *botón*

2. Cuarto elemento hijo de entrada del elemento de formulario con atributo *identificación* empezar a *formulario de inicio de sesión*

Estos ejemplos cubren algunos conceptos básicos, pero para obtener más información, se recomiendan las siguientes referencias:

- Tutorial XPath de W3Schools
- Recomendación XPath del W3C
- Tutorial XPath- con ejemplos interactivos.

Aquí hay un par de complementos muy útiles que pueden ayudar a descubrir el XPath de un elemento:

- Buscador de rutas x- Complemento para obtener los elementos xPath.
- Ayudante XPath- para Google Chrome

4.4 Localización de hipervínculos por texto del enlace

Úselo cuando conozca el texto del enlace utilizado dentro de una etiqueta de anclaje. Con esta estrategia, se devolverá el primer elemento cuyo texto del enlace coincida con el valor proporcionado. Si ningún elemento tiene un atributo de texto de enlace coincidente, se Ninguna excepción de elemento talserá levantado.

Por ejemplo, considere la fuente de esta página:

```
<html>
  <cuerpo>
    <p>¿Estás seguro de que quieres hacer esto?</p>
    <a href="continue.html">Continuar</a> <a
    href="cancel.html">Cancelar</a>
</cuerpo>
</html>
```

El enlace continue.html se puede ubicar así:

```
continuar_enlace=conductor.encontrar_elemento(Por.ENLACE_TEXTO,'Continuar')
continuar_enlace=conductor.encontrar_elemento(Por.PARTIAL_LINK_TEXT,'Conti')
```

4.5 Ubicación de elementos por nombre de etiqueta

Úselo cuando desee ubicar un elemento por nombre de etiqueta. Con esta estrategia, se devolverá el primer elemento con el nombre de etiqueta dado. Si ningún elemento tiene un nombre de etiqueta coincidente, seNinguna excepción de elemento talserá levantado.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <h1>Bienvenido</h1>
    <pag>El contenido del sitio va aquí..</pag>
</cuerpo>
</HTML>
```

El elemento de encabezado (h1) se puede ubicar así:

```
encabezado1=conductor.encontrar_elemento(Por.ETIQUETA_NOMBRE,'h1')
```

4.6 Ubicación de elementos por nombre de clase

Úselo cuando desee ubicar un elemento por nombre de clase. Con esta estrategia, se devolverá el primer elemento con el atributo de nombre de clase coincidente. Si ningún elemento tiene un atributo de nombre de clase coincidente, seNinguna excepción de elemento talserá levantado.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <clase p="contenido">El contenido del sitio va aquí..</pag> </
cuerpo>
</HTML>
```

El elemento “p” se puede ubicar así:

```
contenido=conductor.encontrar_elemento(Por.CLASE_NOMBRE,'contenido')
```

4.7 Localización de elementos mediante selectores CSS

Utilízalo cuando quieras localizar un elemento usandoSelector CSSsintaxis. Con esta estrategia, se devolverá el primer elemento que coincida con el selector CSS dado. Si ningún elemento coincide con el selector CSS proporcionado, seNinguna excepción de elemento tal será levantado.

Por ejemplo, considere la fuente de esta página:

```
<HTML>
  <cuerpo>
    <clase p="contenido">El contenido del sitio va aquí..</pag> </
cuerpo>
</HTML>
```

El elemento “p” se puede ubicar así:

```
contenido=conductor.encontrar_elemento(Por.CSS_SELECTOR,'p.contenido')
```

Sauce Labs tiene buena documentación.en selectores CSS.

CAPÍTULO

CINCO

MURGA

Hoy en día, la mayoría de las aplicaciones web utilizan técnicas AJAX. Cuando el navegador carga una página, los elementos dentro de esa página pueden cargarse en diferentes intervalos de tiempo. Esto dificulta la localización de elementos: si un elemento aún no está presente en el DOM, una función de localización generará un*ElementoNotVisibleException*excepción. Usando esperas, podemos resolver este problema. La espera proporciona cierta holgura entre las acciones realizadas, principalmente localizar un elemento o cualquier otra operación con el elemento.

Selenium Webdriver proporciona dos tipos de esperas: implícitas y explícitas. Una espera explícita hace que WebDriver espere a que se produzca una determinada condición antes de continuar con la ejecución. Una espera implícita hace que WebDriver sondee el DOM durante un cierto período de tiempo al intentar localizar un elemento.

5.1 Esperas explícitas

Una espera explícita es un código que usted define para esperar a que ocurra una determinada condición antes de continuar con el código. El caso extremo de esto es time.sleep(), que establece la condición en un período de tiempo exacto para esperar. Se proporcionan algunos métodos convenientes que lo ayudarán a escribir código que esperará solo el tiempo necesario. WebDriverWait en combinación con ExpectedCondition es una forma de lograrlo.

```
deselenioimportarcontrolador web
deselenio.webdriver.common.byimportarPor
deselenium.webdriver.support.esperarimportarWebDriverEsperar de
selenio.webdriver.soporteimportarcondiciones_esperadascomoCE

conductor=controlador web.Firefox()
conductor.conseguir("http://algundominio/url_que_retrasa_la carga")
intentar:
    elemento=WebDriverWait(controlador,10).hasta(
        CE.presencia_de_elemento_ubicado((Por.IDENTIFICACIÓN,"miElementoDinámico"))
    )
finalmente:
    conductor.abandonar()
```

En el código anterior, Selenium esperará un máximo de 10 segundos para encontrar un elemento que coincida con los criterios dados. Si no se encuentra ningún elemento en ese tiempo, se lanza una TimeoutException. De forma predeterminada, WebDriverWait llama a ExpectedCondition cada 500 milisegundos hasta que devuelve el éxito. La condición esperada volverá *verdadero*(Booleano) en caso de éxito o*no nulo*si no logra localizar un elemento.

Condiciones esperadas

Existen algunas condiciones comunes que se utilizan con frecuencia al automatizar los navegadores web. A continuación se enumeran los nombres de cada uno. El enlace de Selenium Python proporciona algunosmétodos de conveniencia por lo que no tiene que codificar una clase de condición_esperada usted mismo ni crear su propio paquete de utilidades para ellos.

- título_es
- título_contiene
- presencia_de_elemento_ubicado
- visibilidad_del_elemento_ubicado
- visibilidad_de
- presencia_de_todos_los_elementos_ubicados
- text_to_be_present_in_element
- text_to_be_present_in_element_value
- frame_to_be_available_and_switch_to_it
- invisibilidad_del_elemento_ubicado
- element_to_be_clickable
- estancamiento_de
- elemento_a_ser_seleccionado
- elemento_ubicado_a_ser_seleccionado
- elemento_selección_estado_a_ser
- elemento_ubicado_selección_estado_a_ser
- alerta_está_presente

```
deselenio.webdriver.soporteimportarcondiciones_esperadascomoCE

esperar=WebDriverWait(controlador,10)
elemento=esperar.hasta(CE.element_to_be_clickable((Por.IDENTIFICACIÓN,'alguien')))
```

El módulo expected_conditions contiene un conjunto de condiciones predefinidas para usar con WebDriverWait.

Condiciones de espera personalizadas

También puede crear condiciones de espera personalizadas cuando ninguno de los métodos de conveniencia anteriores se ajuste a sus necesidades. Se puede crear una condición de espera personalizada usando una clase con*__llamar__*método que devuelve*FALSO*cuando la condición no coincide.

```
claseelemento_has_css_class(objeto):
    """Una expectativa para comprobar que un elemento tiene una clase CSS particular.

    localizador - usado para encontrar el elemento
    devuelve el WebElement una vez que tiene la clase CSS particular """

    definición__inicio__(ser, localizador, css_class):
        ser.locador=locador ser.
        clase_css=clase_css

    definición__llamar__(ser, conductor):
        elemento=conductor.encontrar_elemento(*ser.locador)#Encontrar el elemento referenciado
        clase_cssenelemento.obtener_atributo("clase"):
            devolverelemento
        demás:
            devolver falso
```

(continúa en la página siguiente)

(continúa de la página anterior)

```
# Esperar hasta que un elemento con id='myNewInput' tenga la clase 'myCSSClass'
esperar=WebDriverWait(controlador,10)
elemento=esperar.hasta(element_has_css_class((Por.IDENTIFICACIÓN,'miNuevaEntrada'),"miClaseCSS"))
```

Nota: Biblioteca polling2

También puedes considerar usarsondeo2biblioteca que debe instalar por separado.

5.2 Esperas implícitas

Una espera implícita le dice a WebDriver que sondee el DOM durante un cierto período de tiempo cuando intenta encontrar cualquier elemento (o elementos) que no esté disponible de inmediato. La configuración predeterminada es 0 (cero). Una vez configurada, la espera implícita se establece durante la vida del objeto WebDriver.

```
deselenioimportarcontrolador web

conductor=controlador web.Controlador Firefox().
implícitamente_esperar(10)#artículos de segunda clase
conductor.conseguir("http://algundominio/url_que_retrasa_la carga")
miElementoDinámico=conductor.buscar_elemento_por_id("miElementoDinámico")
```

CAPÍTULO

SEIS

OBJETOS DE PÁGINA

Este capítulo es una introducción tutorial al patrón de diseño de objetos de página. Un objeto de página representa un área donde la prueba interactúa dentro de la interfaz de usuario de la aplicación web.

Beneficios de usar el patrón de objeto de página:

- Casos de prueba fáciles de leer
- Crear código reutilizable que pueda compartirse en múltiples casos de prueba.
- Reducir la cantidad de código duplicado
- Si la interfaz de usuario cambia, la solución necesita cambios en un solo lugar

6.1 Caso de prueba

Aquí hay un caso de prueba que busca una palabra en el*python.org*sitio web y garantiza algunos resultados. La siguiente sección presentará la*página*módulo donde se definirán los objetos de la página.

```
importarprueba unitaria
deseleniomportarcontrolador web
importarpágina

clasePythonOrgSearch(prueba unitaria.Caso de prueba):
    """Una clase de prueba de muestra para mostrar cómo funciona el objeto de página"""

    definiciónconfiguración(ser):
        ser.conductor=controlador web.Firefox() ser.
        conductor.conseguir("http://www.python.org")

    definiciónprueba_búsqueda_en_python_org(ser):
        """Prueba la función de búsqueda de python.org. Busca la palabra "pycon" y luego
        verifica que aparecen algunos resultados. Tenga en cuenta que no busca ningún texto
        en particular en la página de resultados de búsqueda. Esta prueba verifica que los
        resultados no estén vacíos. """

        # Cargar la página principal. En este caso la página de inicio de Python.org.
        página_principal=página.Página principal(ser.conductor)
        # Comprueba si la palabra "Python" está en el título
        ser.afirmarVerdadero(página_principal.is_title_matches(),"El título de python.org no coincide".)
        # Establece el texto del cuadro de texto de búsqueda en "pycon"
        página_principal.elemento_texto_búsqueda="pycon"
```

(continúa en la página siguiente)

(continúa de la página anterior)

```
            página_principal.click_go_button()
            página_resultados_búsqueda=página.Página de resultados de búsqueda (ser.conductor)
            # Verifica que la página de resultados no esté vacía
            ser.afirmarVerdadero(página_resultados_búsqueda.is_results_found(),"No se encontraron resultados".)

        definicióndemoler(ser):
            ser.conductor.cerca()

si__nombre__=="__principal__":
    prueba unitaria.principal()
```

6.2 Clases de objetos de página

El patrón de objeto de página pretende crear un objeto para cada parte de una página web. Esta técnica ayuda a crear una separación entre el código de prueba y el código real que interactúa con la página web.

Elpágina.pyse verá así:

```
deelementoimportarElemento de página base de
localizadoresimportarLocalizadores de página principal

claseElemento de texto de búsqueda(Elemento de página base):
    """Esta clase obtiene el texto de búsqueda del localizador especificado"""

    # El localizador del cuadro de búsqueda donde se ingresa la cadena de búsqueda
    locador='q'


clasePágina base(objeto):
    """Clase base para inicializar la página base que será llamada desde todas las
    páginas"""

    definición__inicio__(ser, conductor):
        ser.conductor=conductor


clasePágina principal(Página base):
    """Los métodos de acción de la página de inicio vienen aquí. Es decir, Python.org"""

    # Declara una variable que contendrá el texto recuperado
    elemento_texto_búsqueda=Elemento de texto de búsqueda()

    definiciónis_title_matches(ser):
        """Verifica que el texto codificado "Python" aparece en el título de la página"""

        devolver"Pitón"enser.conductor.título

    definiciónclick_go_button(ser):
        """Activa la búsqueda"""

        elemento=ser.conductor.encontrar_elemento(*Localizadores de página principal.GO_BUTTON)
```

(continúa en la página siguiente)

(continúa de la página anterior)

```
            elemento.hacer clic()


clasePágina de resultados de búsqueda(Página base):
    """Los métodos de acción de la página de resultados de búsqueda vienen aquí"""

    definiciónse_resultados_encontrados(ser):
        # Probablemente debería buscar este texto en la página específica
        # elemento, pero por ahora funciona bien
        devolver"No se encontraron resultados."no enser.conductor.fuente_página
```

6.3 Elementos de la página

Elelemento.pyse verá así:

```
deselenio.webdriver.common.byimportarPor
deselenio.webdriver.support.uiimportarWebDriverEsperar


claseElemento de página base(objeto):
    """Clase de página base que se inicializa en cada clase de objeto de página."""

    definición__colocar__(ser, objeto, valor):
        """Establece el texto al valor proporcionado"""

        conductor=objeto.controlador
        WebDriverWait(controlador,100).hasta(
            lambdaconductor: conductor.encontrar_elemento(Por.NOMBRE,ser.localizador))
        conductor.encontrar_elemento(Por.NOMBRE,ser.locador).controlador claro().
        encontrar_elemento(Por.NOMBRE,ser.locador).enviar_claves (valor)

    definición__conseguir__(ser, obj, propietario):
        """Obtiene el texto del objeto especificado"""

        conductor=objeto.controlador
        WebDriverWait(controlador,100).hasta(
            lambdaconductor: conductor.encontrar_elemento(Por.NOMBRE,ser.localizador))
        elemento=conductor.encontrar_elemento(Por.NOMBRE,ser.locador) devolverelemento
        .obtener_atributo("valor")
```

6.4 Localizadores

Una de las prácticas es separar las cuerdas localizadoras del lugar donde se están utilizando. En este ejemplo, los localizadores de la misma página pertenecen a la misma clase.

Ellocalizadores.pyse verá así:

```
deselenio.webdriver.common.byimportarPor

claseLocalizadores de página principal(objeto):
    """Una clase para localizadores de páginas principales. Todos los localizadores de páginas principales deben venir aquí"""

    GO_BUTTON=(Por.IDENTIFICACIÓN,'entregar')

claseLocalizadores de páginas de resultados de búsqueda(objeto):
    """Una clase para localizadores de resultados de búsqueda. Todos los localizadores de resultados de búsqueda deben
    venir aquí"""

    aprobar
```

CAPÍTULO

SIETE

API DE CONDUCTOR WEB

Nota: Esta no es una documentación oficial. La documentación oficial de la API está disponibleaquí.

Este capítulo cubre todas las interfaces de Selenium WebDriver. **Estilo de**

importación recomendado

Las definiciones de API en este capítulo muestran la ubicación absoluta de las clases. Sin embargo, el estilo de importación recomendado es el siguiente:

```
deselenioimportarcontrolador web
```

Luego, podrás acceder a las clases así:

```
controlador web.Firefox
controlador web.Controlador web
FirefoxProfile.Controlador web
FirefoxOptions.Controlador web
FirefoxService.Cromo
controlador web.Controlador web
ChromeOptions.Controlador web
ChromeService.Es decir
controlador web.Es decir, Opciones
controlador web.Es decir, servicio
controlador web.Borde
controlador web.Controlador web
ChromiumEdge.Controlador web
EdgeOptions.Controlador web
EdgeService.Safari
controlador web.Controlador web
SafariOptions.Controlador web
SafariService.WebKitGTK
controlador web.Controlador web
WebKitGTKOptions.Controlador web
WebKitGTKService.WPEWebKit
controlador web.Controlador web
WPEWebKitOptions.Controlador web
WPEWebKitService.Remoto
controlador web.Controlador web
DesiredCapabilities.Cadenas de acción
```

(continúa en la página siguiente)

(continúa de la página anterior)

```
controlador web.Apoderado
controlador web.Llaves
```

La clase de claves especiales (Llaves)se puede importar así:

```
deselenium.webdriver.common.keysimportarLlaves
```

Las clases de excepción se pueden importar así (Reemplace elEl nombre de la clase de excepciónconcon el nombre de clase real que se proporciona a continuación):

```
deselenio.excepciones.comunesimportar[ElNombreDeLaClaseExcepción]
```

Convenciones utilizadas en la API

Algunos atributos son invocables (o métodos) y otros no son invocables (propiedades). Todos los atributos invocables terminan con corchetes.

A continuación se muestra un ejemplo de propiedad:

- URL_actual

  URL de la página cargada actualmente.

  Uso:

  ```
  conductor.URL_actual
  ```

A continuación se muestra un ejemplo de un método:

- cerca()

  Cierra la ventana actual.

  Uso:

  ```
  conductor.cerca()
  ```

7.1 Excepciones

Excepciones que pueden ocurrir en todo el código del webdriver.

excepciónselenio.excepciones.comunes.ElementoClickInterceptedException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno,seguimiento de pila: Secuencia[cadena] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

El comando Clic en elemento no se pudo completar porque el elemento que recibe los eventos oscurece el elemento en el que se solicitó hacer clic.

excepciónselenio.excepciones.comunes.ElementoNotInteractableException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción de estado de elemento no válido*

Se lanza cuando un elemento está presente en el DOM pero las interacciones con ese elemento afectarán a otro elemento debido al orden de pintura.

excepciónselenio.excepciones.comunes.ElementoNotSelectableException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción de estado de elemento no válido*

Se produce al intentar seleccionar un elemento no seleccionable.

Por ejemplo, seleccionando un elemento 'script'.

excepciónselenio.excepciones.comunes.ElementoNotVisibleException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción de estado de elemento no válido*

Se lanza cuando un elemento está presente en el DOM, pero no es visible y, por lo tanto, no se puede interactuar con él.

Se produce con mayor frecuencia al intentar hacer clic o leer el texto de un elemento que está oculto a la vista.

excepciónselenio.excepciones.comunes.ImeActivationFailedException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se produce cuando falla la activación de un motor IME.

excepciónselenio.excepciones.comunes.ImeNotAvailableException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se produce cuando la compatibilidad con IME no está disponible.

Esta excepción se produce para cada llamada a un método relacionado con IME si la compatibilidad con IME no está disponible en la máquina.

excepciónselenio.excepciones.comunes.InsecureCertificateException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

La navegación hizo que el agente de usuario apareciera una advertencia de certificado, que generalmente es el resultado de un certificado TLS caducado o no válido.

excepciónselenio.excepciones.comunes.Excepción de argumento no válido (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Los argumentos pasados a un comando no son válidos o están mal formados.

excepciónselenio.excepciones.comunes.Excepción de dominio de cookie no válida (*mensaje: cadena | Ninguno = Ninguno, pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se produce al intentar agregar una cookie en un dominio diferente al de la URL actual.

excepciónselenio.excepciones.comunes.Excepción de coordenadas no válidas (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Las coordenadas proporcionadas para la operación de una interacción no son válidas.

excepciónselenio.excepciones.comunes.InvalidElementStateException(*mensaje: cadena | Ninguno = Ninguno, pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se lanza cuando no se pudo completar un comando porque el elemento está en un estado no

válido. Esto puede deberse al intento de borrar un elemento que no es editable ni reiniciable.

excepciónselenio.excepciones.comunes.Excepción de selección no válida (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se lanza cuando el selector que se utiliza para buscar un elemento no devuelve un WebElement.

Actualmente, esto sólo sucede cuando el selector es una expresión xpath y es sintácticamente inválido (es decir, no es una expresión xpath) o la expresión no selecciona WebElements (por ejemplo, “count(//input)”).

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*) → Ninguno

excepciónselenio.excepciones.comunes.InvalidSessionIdException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Ocurre si la identificación de sesión dada no está en la lista de sesiones activas, lo que significa que la sesión no existe o no está activa.

excepciónselenio.excepciones.comunes.InvalidSwitchToTargetException(*mensaje: cadena | Ninguno = Ninguno, pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se lanza cuando el marco o la ventana que se va a cambiar no existe.

excepciónselenio.excepciones.comunes.Excepción Javascript(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se produjo un error al ejecutar JavaScript proporcionado por el usuario.

excepciónselenio.excepciones.comunes.MoveTargetOutOfBoundsException(*mensaje: cadena | Ninguno = Ninguno*, *pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Lanzado cuando el objetivo proporcionado al*AccionesCadenas*El método move() no es válido, es decir, está fuera del documento.

excepciónselenio.excepciones.comunes.NoAlertPresentException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se produce al cambiar a ninguna alerta presentada.

Esto puede deberse a que se llama a una operación en la clase Alert() cuando aún no aparece una alerta en la pantalla.

excepciónselenio.excepciones.comunes.NoSuchAttributeException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se lanza cuando no se puede encontrar el atributo del elemento.

Es posible que desees comprobar si el atributo existe en el navegador concreto con el que estás realizando la prueba. Algunos navegadores pueden tener diferentes nombres de propiedad para la misma propiedad. (.innerText de IE8 frente a .textContent de Firefox)

excepciónselenio.excepciones.comunes.Ninguna excepción de cookie (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

No se encontró ninguna cookie que coincida con el nombre de la ruta proporcionada entre las cookies asociadas del documento activo del contexto de navegación actual.

excepciónselenio.excepciones.comunes.NosuchDriverException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se genera cuando el controlador no está especificado y no se puede localizar.

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)→ Ninguno

excepciónselenio.excepciones.comunes.NoSuchElementException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web* Lanzado cuando

no se pudo encontrar el elemento.

Si encuentra esta excepción, es posible que desee comprobar lo siguiente:

- Verifique el selector utilizado en su find_by. . .
- Es posible que el elemento aún no esté en la pantalla en el momento de la operación de búsqueda (la página web aún se está cargando). Consulte selenium.webdriver.support.wait.WebDriverWait() para saber cómo escribir un contenedor de espera para esperar a que aparezca un elemento.

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*) → Ninguno

excepciónselenio.excepciones.comunes.NoSuchFrameException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción SwitchToTarget no válida*

Se lanza cuando el objetivo del fotograma que se va a cambiar no existe.

excepciónselenio.excepciones.comunes.NoSuchShadowRootException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se produce al intentar acceder a la raíz oculta de un elemento cuando no tiene una raíz oculta adjunta.

excepciónselenio.excepciones.comunes.Ninguna excepción de ventana tal (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción SwitchToTarget no válida*

Se lanza cuando el destino de la ventana que se va a cambiar no existe.

Para encontrar el conjunto actual de identificadores de ventana activos, puede obtener una lista de los identificadores de ventana activos de la siguiente manera:

```
imprimirconductor.manijas_ventana
```

excepciónselenio.excepciones.comunes.Excepción de captura de pantalla (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Una captura de pantalla se hizo imposible.

excepciónselenio.excepciones.comunes.SesiónNotCreatedException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

No se pudo crear una nueva sesión.

excepciónselenio.excepciones.comunes.StaleElementReferenceException(*mensaje: cadena | Ninguno = Ninguno, pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases: *Excepción del controlador web*

Se lanza cuando una referencia a un elemento ahora está "obsoleta".

Obsoleto significa que el elemento ya no aparece en el DOM de la página.

Las posibles causas de StaleElementReferenceException incluyen, entre otras:

- Ya no está en la misma página o es posible que la página se haya actualizado desde que se ubicó el elemento.
- Es posible que el elemento haya sido eliminado y vuelto a agregar a la pantalla desde que fue localizado. Como por ejemplo un elemento que se está reubicando. Esto puede suceder normalmente con un marco de JavaScript cuando se actualizan los valores y se reconstruye el nodo.

• Es posible que el elemento haya estado dentro de un iframe u otro contexto que se actualizó.

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*) → Ninguno

excepciónselenio.excepciones.comunes.Excepción de tiempo de espera (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se lanza cuando un comando no se completa en el tiempo suficiente.

excepciónselenio.excepciones.comunes.Incapaz de establecer una excepción de cookie (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se lanza cuando un controlador no puede configurar una cookie.

excepciónselenio.excepciones.comunes.AlertaInesperadaPresentException(*mensaje: cadena | Ninguno = Ninguno, pantalla: cadena | Ninguno = Ninguno, stacktrace: Secuencia[str] | Ninguno = Ninguno,texto_alerta: cadena | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se lanza cuando ha aparecido una alerta inesperada.

Generalmente surge cuando un modal inesperado impide que el controlador web ejecute comandos.

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno, texto_alerta: cadena | Ninguno = Ninguno*) →Ninguno

excepciónselenio.excepciones.comunes.Excepción de nombre de etiqueta inesperada (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

Se lanza cuando una clase de soporte no obtuvo un elemento web esperado.

excepciónselenio.excepciones.comunes.Excepción de método desconocido (*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:*Excepción del controlador web*

El comando solicitado coincidió con una URL conocida pero no coincidió con ningún método para esa URL.

excepciónselenio.excepciones.comunes.WebDriverException(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*)

Bases:Excepción

Excepción del controlador web base.

__inicio__(*mensaje: cadena | Ninguno = Ninguno,pantalla: cadena | Ninguno = Ninguno,stacktrace: Secuencia[str] | Ninguno = Ninguno*) → Ninguno

7.2 Cadenas de acción

La implementación de ActionChains.

claseselenium.webdriver.common.action_chains.Cadenas de acción(*conductor:*Controlador web,*duración: int = 250*,*dispositivos: lista[CualquierDispositivo] | Ninguno = Ninguno*)

Bases:objeto

ActionChains son una forma de automatizar interacciones de bajo nivel, como movimientos del mouse, acciones del botón del mouse, pulsación de teclas e interacciones del menú contextual. Esto es útil para realizar acciones más complejas como pasar el cursor por encima y arrastrar y soltar.

Generar acciones de usuario.

Cuando llamas a métodos para acciones en el objeto ActionChains, las acciones se almacenan en una cola en el objeto ActionChains. Cuando llamas a perform(), los eventos se activan en el orden en que están en cola.

ActionChains se puede utilizar en un patrón de cadena:

```
menú=conductor.encontrar_elemento(Por.CSS_SELECTOR,".nav")
submenú_oculto=conductor.encontrar_elemento(Por.CSS_SELECTOR,".nav #submenú1")

Cadenas de acción (controlador).mover_a_elemento(menú).haga clic en (submenú_oculto).llevar a cabo()
```

O las acciones se pueden poner en cola una por una y luego realizarlas:

```
menú=conductor.encontrar_elemento(Por.CSS_SELECTOR,".nav")
submenú_oculto=conductor.encontrar_elemento(Por.CSS_SELECTOR,".nav #submenú1")

comportamiento=Acciones de ActionChains
(controlador).acciones
move_to_element(menú).hacer clic en
(submenú_oculto) acciones.llevar a cabo()
```

De cualquier manera, las acciones se realizan en el orden en que se llaman, una tras otra.

__inicio__(*conductor:*Controlador web,*duración: int = 250*,*dispositivos: lista[CualquierDispositivo] | Ninguno = Ninguno*) →Ninguno

Crea una nueva ActionChains.

argumentos

- controlador: la instancia de WebDriver que realiza acciones del usuario.
- duración: anula los 250 ms predeterminados de DEFAULT_MOVE_DURATION en PointerInput

hacer clic(*en_elemento:*Elemento web | *Ninguno = Ninguno*) →*Cadenas de acción*

Hace clic en un elemento.

argumentos

- on_element: El elemento en el que hacer clic. Si no hay ninguno, hace clic en la posición actual del mouse.

hacer clic y mantener presionado (*en_elemento:*Elemento web | *Ninguno = Ninguno*) →*Cadenas de acción*

Mantiene presionado el botón izquierdo del mouse sobre un elemento.

argumentos

- on_element: El elemento sobre el que colocar el mouse. Si no hay ninguno, hace clic en la posición actual del mouse.

contexto_clic(*en_elemento:*Elemento web | *Ninguno = Ninguno*) → *Cadenas de acción*

Realiza un clic contextual (clic derecho) en un elemento.

argumentos

- on_element: el elemento sobre el que hacer clic contextual. Si no hay ninguno, hace clic en la posición actual del mouse.

doble clic (*en_elemento:*Elemento web | *Ninguno = Ninguno*) → *Cadenas de acción*

Hace doble clic en un elemento.

argumentos

- on_element: El elemento sobre el que hacer doble clic. Si no hay ninguno, hace clic en la posición actual del mouse.

arrastrar y soltar (*fuente:*Elemento web, *objetivo:*Elemento web) → *Cadenas de acción*

Mantiene presionado el botón izquierdo del mouse en el elemento de origen, luego se mueve al elemento de destino y suelta el botón del mouse.

argumentos

- fuente: El elemento sobre el que colocar el mouse.
- objetivo: El elemento sobre el que colocar el mouse.

arrastrar y soltar por desplazamiento (*fuente:*Elemento web, *compensación x: int*, *compensación y: int*) → *Cadenas de acción*

Mantiene presionado el botón izquierdo del mouse en el elemento de origen, luego se mueve al desplazamiento de destino y suelta el botón del mouse.

argumentos

- fuente: El elemento sobre el que colocar el mouse.
- xoffset: desplazamiento X al que moverse.
- yoffset: desplazamiento Y al que desplazarse.

tecla_abajo(*valor: cadena*, *elemento:*Elemento web | *Ninguno = Ninguno*) → *Cadenas de acción*

Envía solo una pulsación de tecla, sin soltarla. Sólo debe usarse con teclas modificadoras (Control, Alt y Shift).

argumentos

- valor: la tecla modificadora a enviar. Los valores se definen en *Llaves* clase.
- elemento: El elemento para enviar claves. Si no hay ninguno, envía una clave al elemento enfocado actual.

Ejemplo, presionando ctrl+c:

```
Cadenas de acción (controlador).key_down(Teclas.CONTROL).enviar_claves('do').key_up(Claves.CONTROL).
→llevar a cabo()
```

clave_arriba(*valor: cadena*, *elemento:*Elemento web | *Ninguno = Ninguno*) → *Cadenas de acción*

Libera una tecla modificadora.

argumentos

- valor: la tecla modificadora a enviar. Los valores se definen en la clase Claves.
- elemento: El elemento para enviar claves. Si no hay ninguno, envía una clave al elemento enfocado actual.

Ejemplo, presionando ctrl+c:

```
Cadenas de acción (controlador).key_down(Teclas.CONTROL).enviar_claves('do').key_up(Claves.CONTROL).
→llevar a cabo()
```

mover_por_desplazamiento(*compensación x: int*,*compensación y: int*) → *Cadenas de acción*

Mover el mouse a un desplazamiento desde la posición actual del mouse.

argumentos

- xoffset: desplazamiento X al que moverse, como un entero positivo o negativo.
- yoffset: desplazamiento Y al que desplazarse, como un entero positivo o negativo.

mover_a_elemento(*a_elemento:* Elemento web) → *Cadenas de acción*

Mover el mouse al centro de un elemento.

argumentos

- to_element: el elemento web al que moverse.

move_to_element_with_offset(*a_elemento:* Elemento web,*compensación x: int*,*compensación y: int*) → *Cadenas de acción*

Mueva el mouse un desplazamiento del elemento especificado. Los desplazamientos son relativos al punto central del elemento en la vista.

argumentos

- to_element: el elemento web al que moverse.
- xoffset: desplazamiento X al que moverse, como un entero positivo o negativo.
- yoffset: desplazamiento Y al que desplazarse, como un entero positivo o negativo.

pausa(*segundos: flotar | entero*) → *Cadenas de acción*

Pausa todas las entradas durante el tiempo especificado en segundos.

llevar a cabo() → Ninguno

Realiza todas las acciones almacenadas.

liberar(*en_elemento:* Elemento web | *Ninguno = Ninguno*) → *Cadenas de acción*

Soltar un botón del mouse presionado sobre un elemento.

argumentos

- on_element: El elemento sobre el que colocar el mouse. Si no hay ninguno, se suelta en la posición actual del mouse.

restablecer_acciones() → Ninguno

Borra acciones que ya están almacenadas localmente y en el extremo remoto.

desplazamiento_por_cantidad(*delta_x:int*,*delta_y:int*) → *Cadenas de acción*

Se desplaza por las cantidades proporcionadas con el origen en la esquina superior izquierda de la ventana gráfica.

argumentos

- delta_x: Distancia a lo largo del eje X para desplazarse mediante la rueda. Un valor negativo se desplaza hacia la izquierda.
- delta_y: Distancia a lo largo del eje Y para desplazarse usando la rueda. Un valor negativo se desplaza hacia arriba.

desplazamiento_desde_origen(*scroll_origin: OrigenDesplazamiento*,*delta_x:int*,*delta_y:int*) → *Cadenas de acción*

Se desplaza por la cantidad proporcionada según el origen proporcionado. El origen del desplazamiento es el centro de un elemento o la parte superior izquierda de la ventana gráfica más cualquier desplazamiento. Si el origen es un elemento y el elemento no está en la ventana gráfica, la parte inferior del elemento primero se desplazará hasta la parte inferior de la ventana gráfica.

argumentos

- origen: donde se origina el desplazamiento (ventana gráfica o centro del elemento) más los desplazamientos proporcionados.
- delta_x: Distancia a lo largo del eje X para desplazarse mediante la rueda. Un valor negativo se desplaza hacia la izquierda.
- delta_y: Distancia a lo largo del eje Y para desplazarse usando la rueda. Un valor negativo se desplaza hacia arriba.

aumentos

Si el origen con desplazamiento está fuera de la ventana gráfica. -Excepción MoveTargetOutOfBounds

- Si el origen con desplazamiento está fuera de la ventana gráfica.

desplazamiento_a_elemento(*elemento:* Elemento web) → *Cadenas de acción*

Si el elemento está fuera de la ventana gráfica, desplaza la parte inferior del elemento hasta la parte inferior de la ventana gráfica.

argumentos

- elemento: qué elemento desplazarse en la ventana gráfica.

enviar_claves(**keys_to_send: str*) → *Cadenas de acción*

Envía claves al elemento enfocado actual.

argumentos

- llaves_to_send: Las claves a enviar. Las constantes de las claves modificadoras se pueden encontrar en la clase 'Claves'.

enviar_claves_al_elemento(*elemento:* Elemento web, **claves_para_enviar: cadena*) → *Cadenas de acción*

Envía claves a un elemento.

argumentos

- elemento: El elemento para enviar claves.
- llaves_to_send: Las claves a enviar. Las constantes de las claves modificadoras se pueden encontrar en la clase 'Claves'.

7.3 Alertas

La implementación de Alerta.

claseselenium.webdriver.common.alert.Alerta(*conductor*)

Bases:objeto

Permite trabajar con alertas.

Utilice esta clase para interactuar con mensajes de alerta. Contiene métodos para descartar, aceptar, ingresar y obtener texto de mensajes de alerta.

Aceptar o descartar mensajes de alerta:

```
Alerta (conductor).aceptar()
Alerta(conductor).despedir()
```

Ingresando un valor en un mensaje de alerta:

```
nombre_prompt=Alerta (conductor)
nombre_prompt.enviar_claves("Willian Shakesphere")
nombre_prompt.aceptar()
```

Lectura del texto de una solicitud de verificación:

```
texto_alerta=Alerta (conductor).texto
ser.afirmarIgual("¿Quieres dejar de fumar?", texto_alerta)
```

__inicio__(*conductor*) → Ninguno

Crea una nueva alerta.

argumentos

• controlador: la instancia de WebDriver que realiza acciones del usuario.

aceptar() →Ninguno

Acepta la alerta disponible.

Uso

```
Alerta (conductor).aceptar()#Confirmar un diálogo de alerta.
```

despedir() →Ninguno

Descarta la alerta disponible.

enviar_claves(*claves para enviar: str*) →Ninguno

Enviar claves a la alerta.

argumentos

•keysToSend: El texto que se enviará a Alerta.

texto de propiedad: str

Obtiene el texto de la alerta.

7.4 Teclas especiales

La implementación de las claves.

claseselenium.webdriver.common.keys.Llaves

Bases:objeto

Conjunto de códigos de claves especiales.

AÑADIR = '\ue025'

ALT = '\ue00a'

ARROW_DOWN = '\ue015'

ARROW_LEFT = '\ue012'

ARROW_RIGHT = '\ue014'

FLECHA_UP = '\ue013'

RETROCESO = '\ue003'

ESPACIO_RETROCESO = '\ue003'

CANCELAR = '\ue001'

BORRAR = '\ue005'

COMANDO = '\ue03d'

CONTROL = '\ue009'

DECIMAL = '\ue028'

BORRAR = '\ue017'

DIVIDIR = '\ue029'

ABAJO = '\ue015'

FINAL = '\ue010'

ENTRAR = '\ue007'

IGUAL = '\ue019'

ESCAPAR = '\ue00c'

F1 = '\ue031'

F10 = '\ue03a'

F11 = '\ue03b'

F12 = '\ue03c'

F2 = '\ue032'

F3 = '\ue033'

F4 = '\ue034'

F5 = '\ue035'

F6 = '\ue036'

F7 = '\ue037'

F8 = '\ue038'

F9 = '\ue039'

AYUDA = '\ue002'

INICIO = '\ue011'

INSERTAR = '\ue016'

IZQUIERDA = '\ue012'

LEFT_ALT = '\ue00a'

LEFT_CONTROL = '\ue009'

LEFT_SHIFT = '\ue008'

META = '\ue03d'

MULTIPLICAR = '\ue024'

NULO = '\ue000'

NUMPAD0 = '\ue01a'

TECL NUM1 = '\ue01b'

TECL NUM2 = '\ue01c'

TECL NUM3 = '\ue01d'

TECL NUM4 = '\ue01e'

TECL NUM5 = '\ue01f'

TECL NUM6 = '\ue020'

TECL NUM7 = '\ue021'

TECL NUM8 = '\ue022'

TECL NUM9 = '\ue023'

PÁGINA_ABAJO = '\ue00f'

PÁGINA_ARRIBA = '\ue00e'

PAUSA = '\ue00b'

RETORNO = '\ue006'

DERECHA = '\ue014'

PUNTO Y COMA = '\ue018'

SEPARADOR = '\ue026'

MAYÚS = '\ue008'

ESPACIO = '\ue00d'

RESTA = '\ue027'

TAB = '\ue004'

ARRIBA = '\ue013'

ZENKAKU_HANKAKU = '\ue040'

7.5 Localizar elementos por

Estos son los atributos que se pueden utilizar para localizar elementos. Ver el*Localización de elementos*capítulo, por ejemplo, usos. La implementación Por.

claseselenium.webdriver.common.by.Por

Bases:objeto

Conjunto de estrategias de localización soportadas.

CLASS_NAME = 'nombre de clase'

CSS_SELECTOR = 'selector css'

identificación = 'identificación'

LINK_TEXT = 'texto del enlace'

NOMBRE = 'nombre'

PARTIAL_LINK_TEXT = 'texto de enlace parcial'

TAG_NAME = 'nombre de etiqueta'

XPATH = 'xruta'

7.6 Capacidades Deseadas

Ver el*Usando Selenium con WebDriver remoto*sección, por ejemplo, usos de las capacidades deseadas. el deseado Implementación de capacidades.

claseselenium.webdriver.common.desired_capabilities.Capacidades deseadas

Bases:objeto

Conjunto de capacidades deseadas admitidas de forma predeterminada.

Utilice esto como punto de partida para crear un objeto de capacidades deseado para solicitar controladores web remotos para conectarse al servidor Selenium o a la red Selenium.

Ejemplo de uso:

```
deselenioimportarcontrolador web

selenio_grid_url="http://198.0.0.1:4444/wd/hub"

# Cree un objeto de capacidades deseado como punto de partida.
capacidades=Capacidades deseadas.FIREFOX.capacidades de copia()[
'plataforma']="VENTANAS" capacidades['versión']="10"


# Cree una instancia de Remote WebDriver con las capacidades deseadas. conductor=
controlador web.Remoto(capacidades_deseado=capacidades,
                                  comando_ejecutor=selenio_grid_url)
```

Nota: Utilice siempre '.copy()' en el objeto DesiredCapabilities para evitar los efectos secundarios de alterar la instancia de clase Global.

CROMO = {'nombre del navegador': 'cromo'}

BORDE = {'nombre del navegador': 'MicrosoftEdge'}

FIREFOX = {'acceptInsecureCerts': Verdadero, 'browserName': 'firefox',
'moz:debuggerAddress': Verdadero}

HTMLUNIT = {'browserName': 'htmlunit', 'plataforma': 'CUALQUIER', 'versión': ''}

HTMLUNITWITHJS = {'browserName': 'htmlunit', 'javascriptEnabled': Verdadero,
'plataforma': 'CUALQUIERA', 'versión': 'firefox'}

INTERNETEXPLORER = {'nombredelnavegador': 'internet explorer', 'nombre de plataforma': 'windows'}

IPAD = {'nombre del navegador': 'iPad', 'plataforma': 'mac', 'versión': ''}

IPHONE = {'nombre del navegador': 'iPhone', 'plataforma': 'mac', 'versión': ''}

SAFARI = {'nombre del navegador': 'safari', 'nombre de la plataforma': 'impermeable'}

WEBKITGTK = {'nombre del navegador': 'MiniNavegador'}

WPEWEBKIT = {'nombre del navegador': 'MiniNavegador'}

7.7 Representante

La implementación del proxy.

claseselenium.webdriver.common.proxy.Apoderado(*crudo=Ninguno*)

Bases:objeto

Proxy contiene información sobre el tipo de proxy y la configuración de proxy necesaria.

__inicio__(*crudo=Ninguno*)

Crea un nuevo proxy.

argumentos

• sin procesar: datos de proxy sin procesar. Si es Ninguno, se utilizan los valores de clase predeterminados.

to_capabilities()

detección_automática

Obtiene y establece*detección_automática*

7.7.1 Uso

- **Conseguir**
  - *self.auto_detectar*
- **Colocar**
  - *self.auto_detectar=valor*

7.7.2 Parámetros

valor:*cadena*

detección automática = falso

ftpProxy = ''

ftp_proxy

Obtiene y establece*ftp_proxy*

7.7.3 Uso

- Conseguir
  - *self.ftp_proxy*
- Colocar
  - *self.ftp_proxy=valor*

7.7.4 Parámetros

valor:*cadena*

httpProxy = ''

http_proxy

Obtiene y establece*http_proxy*

7.7.5 Uso

- Conseguir
  - *self.http_proxy*
- Colocar
  - *self.http_proxy=valor*

7.7.6 Parámetros

valor:*cadena*

noProxy = ''

no_proxy

Obtiene y establece*no_proxy*

7.7.7 Uso

- Conseguir
  - *self.no_proxy*
- Colocar
  - *self.no_proxy=valor*

7.7.8 Parámetros

valor:*cadena*

proxyAutoconfigUrl = ''

tipo de proxy = {'ff_value': 6, 'cadena': 'NO ESPECIFICADO'}

proxy_autoconfig_url

Obtiene y establece*proxy_autoconfig_url*

7.7.9 Uso

- **Conseguir**
  - *self.proxy_autoconfig_url*
- **Colocar**
  - *self.proxy_autoconfig_url=valor*

7.7.10 Parámetro

valor:*cadena*

propiedad tipo_proxy

Devuelve el tipo de proxy como*Tipo de proxy.*

calcetinesContraseña = ''

calcetinesProxy = ''

calcetinesNombre de usuario = ''

calcetinesVersión = Ninguno

calcetines_contraseña

Obtiene y establece*calcetines_contraseña*

7.7.11 Uso

- **Conseguir**
  - *self.socks_contraseña*
- **Colocar**
  - *self.socks_contraseña=valor*

7.7.12 Parámetro

valor:*cadena*

calcetines_proxy

Obtiene y establece*calcetines_proxy*

7.7.13 Uso

- **Conseguir**
  - *self.sock_proxy*
- **Colocar**
  - *self.socks_proxy=valor*

7.7.14 Parámetro

valor:*cadena*

calcetines_nombre de usuario

Obtiene y establece*calcetines_contraseña*

7.7.15 Uso

- **Conseguir**
  - *self.socks_contraseña*
- **Colocar**
  - *self.socks_contraseña=valor*

7.7.16 Parámetro

valor:*cadena*

versión_calcetines

Obtiene y establece*versión_calcetines*

7.7.17 Uso

- **Conseguir**
  - *self.socks_version*
- **Colocar**
  - *self.socks_version=valor*

7.7.18 Parámetro

valor:*cadena*

SSLProxy = ''

proxy_ssl

Obtiene y establece*proxy_ssl*

7.7.19 Uso

- **Conseguir**
  - *self.ssl_proxy*
- **Colocar**
  - *self.ssl_proxy=valor*

7.7.20 Parámetro

valor:*cadena*

claseselenium.webdriver.common.proxy.Tipo de proxy

Bases:objeto

Conjunto de posibles tipos de proxy.

Cada tipo de proxy tiene 2 propiedades: 'ff_value' es el valor de la preferencia del perfil de Firefox, 'string' es la identificación del tipo de proxy.

carga del método de clase (*valor*)

AUTODETECCIÓN = {'ff_value': 4, 'cadena': 'AUTODETECCIÓN'}

DIRECTO = {'ff_value': 0, 'cadena': 'DIRECTO'}

MANUAL = {'ff_valor': 1, 'cadena': 'MANUAL'}

PAC = {'ff_value': 2, 'cadena': 'PAC'}

RESERVADO_1 = {'ff_value': 3, 'cadena': 'RESERVADO1'}

SISTEMA = {'ff_value': 5, 'cadena': 'SISTEMA'}

NO ESPECIFICADO = {'ff_value': 6, 'cadena': 'NO ESPECIFICADO'}

claseselenium.webdriver.common.proxy.Fábrica de tipo de proxy

Bases:objeto

Fábrica de tipos de proxy. marca

estática (*valor_ff*,*cadena*)

7.8 Utilidades

Los métodos de utilidades.

selenium.webdriver.common.utils.encontrar_ip_conectable(*anfitrión: cadena | bytes | matriz de bytes | Ninguno*,*puerto: int | Ninguno = Ninguno*) →cadena | Ninguno

Resuelva un nombre de host en una IP, prefiriendo direcciones IPv4.

Preferimos IPv4 para no cambiar el comportamiento de implementaciones anteriores solo de IPv4 y porque algunos controladores (por ejemplo, FirefoxDriver) no admiten conexiones IPv6.

Si se proporciona el número de puerto opcional, solo se consideran las IP que escuchan en el puerto determinado.

argumentos

- host: un nombre de host.
- puerto: número de puerto opcional.

Devoluciones

Una única dirección IP, como una cadena. Si se encuentra alguna dirección IPv4, se devuelve una. De lo contrario, si se encuentra alguna dirección IPv6, se devuelve una. En caso contrario, se devuelve Ninguno.

selenium.webdriver.common.utils.puerto_libre() →entero

Determina un puerto libre mediante sockets.

selenium.webdriver.common.utils.es_conectable(*puerto: int*,*anfitrión: cadena | Ninguno = 'localhost'*) →booleano

Intenta conectarse al servidor en el puerto para ver si se está ejecutando.

argumentos

- puerto: el puerto al que conectarse.

selenium.webdriver.common.utils.es_url_conectable(*puerto: int | cadena*) →booleano

Intenta conectarse al servidor HTTP en la ruta /status y el puerto especificado para ver si responde correctamente.

argumentos

- puerto: el puerto al que conectarse.

selenium.webdriver.common.utils.join_host_port(*anfitrión: cadena*,*puerto: int*) →cadena

Une un nombre de host y un puerto.

Esta es una implementación mínima destinada a hacer frente a los literales de IPv6. Por ejemplo, _join_host_port('::1', 80) == '[::1]:80'.

argumentos

- host: un nombre de host.
- puerto: un puerto entero.

selenium.webdriver.common.utils.teclas_para_escribir(*valor: Iterable[cadena | entero | flotar]*) →Lista[cadena]

Procesa los valores que se escribirán en el elemento.

7.9 Servicio

claseselenium.webdriver.common.service.Servicio(*ruta_ejecutable: str | Ninguno = Ninguno,puerto: int = 0*,
*salida_log: int | cadena | IO[Cualquiera] | Ninguno = Ninguno,env: Mapeo[Cualquiera, Cualquiera] | Ninguno = Ninguno,**kwargs*)

Bases:abecedario

La clase base abstracta para todos los objetos de servicio. Los servicios suelen iniciar un programa secundario en un nuevo proceso como proceso provisional para comunicarse con un navegador.

Parámetros

- ejecutable –ruta de instalación del ejecutable.
- puerto -El puerto en el que se ejecutará el servicio tiene por defecto 0, donde el sistema operativo decidirá.
- salida_log – (Opcional) representación int de STDOUT/DEVNULL, cualquier instancia de IO o ruta de cadena al archivo.
- entorno – (Opcional) Mapeo de variables de entorno para el nuevo proceso, el valor predeterminado es *os.environ*.

__inicio__(*ruta_ejecutable: str | Ninguno = Ninguno,puerto: int = 0,salida_log: int | cadena | IO[Cualquiera] | Ninguno = Ninguno,*
*env: Mapeo[Cualquiera, Cualquiera] | Ninguno = Ninguno,**kwargs*) →Ninguno

afirmar_proceso_still_running() →Ninguno

Compruebe si el proceso subyacente todavía se está ejecutando.

resumen línea_comando_args() →Lista[cadena]

Una lista de argumentos del programa (excluyendo el ejecutable).

es_conectable() →booleano

Establece una conexión de socket para determinar si se puede acceder al servicio que se ejecuta en el puerto.

send_remote_shutdown_command() →Ninguno

Envíe una solicitud HTTP al punto final de cierre del servicio en un intento de detenerlo.

comenzar() →Ninguno

Inicia el Servicio.

Excepciones

- WebDriverException: se genera cuando no se puede iniciar el servicio o cuando no se puede conectar al servicio.

detener() →Ninguno

Detiene el servicio.

ruta de propiedad: str

propiedad service_url: str

Obtiene la URL del Servicio.

7.10 Caché de aplicaciones

7.11 Controlador web de Firefox

claseselenio.webdriver.firefox.webdriver.Controlador web(*opciones:*Opciones | *Ninguno = Ninguno,servicio:* *Servicio | Ninguno = Ninguno,keep_alive: bool = Verdadero*)

Bases: *Controlador web*

Controla el GeckoDriver y le permite manejar el navegador.

__inicio__(*opciones:*Opciones | *Ninguno = Ninguno,servicio: Servicio | Ninguno = Ninguno,keep_alive: bool = Verdadero*) → Ninguno

Crea una nueva instancia del controlador de Firefox. Inicia el servicio y luego crea una nueva instancia del controlador de Firefox.

argumentos

- opciones: instancia deopciones.Opciones.
- servicio: instancia de servicio (opcional) para gestionar el arranque y la parada del conductor.
- keep_alive: si se debe configurar remote_connection.RemoteConnection para utilizar HTTP keep-alive.

contexto(*contexto*)

Establece el contexto en el que se ejecutan los comandos de Selenium usando un *con*declaración. El estado del contexto en el servidor se guarda antes de ingresar al bloque y se restaura al salir del mismo.

Parámetros

contexto -Contexto, puede ser una de las propiedades de la clase.*CONTEXTO_CROMO*o *CONTEXT_CONTENIDO*.

Ejemplo de uso:

```
conselenio.contexto (selenio.CONTEXTO_CHROME):
    # alcance cromado
    . . . hacer cosas...
```

get_full_page_screenshot_as_base64() → cadena

Obtiene la captura de pantalla completa del documento de la ventana actual como una cadena codificada en base64 que es útil en imágenes incrustadas en HTML.

Uso

```
conductor.get_full_page_screenshot_as_base64()
```

get_full_page_screenshot_as_file(*Nombre del archivo*) → booleano

Guarda una captura de pantalla del documento completo de la ventana actual en un archivo de imagen PNG. Devuelve False si hay algún IOError; en caso contrario, devuelve True. Utilice rutas completas en su nombre de archivo.

argumentos

- nombre de archivo: la ruta completa en la que desea guardar su captura de pantalla. Esto debería terminar con un *png* extensión.

Uso

```
conductor.get_full_page_screenshot_as_file('/Capturas de pantalla/foo.png')
```

get_full_page_screenshot_as_png() →bytes

Obtiene la captura de pantalla completa del documento de la ventana actual como datos binarios.

Uso

```
conductor.get_full_page_screenshot_as_png()
```

instalar_addon(*camino*,*temporal=Falso*) →cadena

Instala el complemento Firefox.

Devuelve el identificador del complemento instalado. Este identificador se puede utilizar más adelante para desinstalar el complemento.

Parámetros

- temporario -le permite cargar extensiones del navegador temporalmente durante una sesión
- camino -Ruta absoluta al complemento que se instalará.

Uso

```
conductor.instalar_addon('/ruta/a/firebug.xpi')
```

abandonar() →Ninguno

Cierra el navegador y cierra el ejecutable de GeckoDriver.

save_full_page_screenshot(*Nombre del archivo*) →booleano

Guarda una captura de pantalla del documento completo de la ventana actual en un archivo de imagen PNG. Devuelve False si hay algún IOError; en caso contrario, devuelve True. Utilice rutas completas en su nombre de archivo.

argumentos

• nombre de archivo: la ruta completa en la que desea guardar su captura de pantalla. Esto debería terminar con un.*png* extensión.

Uso

```
conductor.save_full_page_screenshot('/Capturas de pantalla/foo.png')
```

establecer_contexto(*contexto*) →Ninguno

desinstalar_addon(*identificador*) →Ninguno

Desinstala el complemento de Firefox usando su identificador.

Uso

```
conductor.desinstalar_addon(' addon@foo.com ')
```

CONTEXT_CHROME = 'cromo'

CONTEXT_CONTENT = 'contenido'

7.12 Opciones del controlador web de Firefox

claseselenium.webdriver.firefox.opciones.Registro

Bases:objeto

__inicio__() →Ninguno

to_capabilities() →dictar

claseselenium.webdriver.firefox.opciones.Opciones

Bases:OpcionesArg

__inicio__() →Ninguno

habilitar_móvil(*android_package: str = 'org.mozilla.firefox'*,*android_actividad=Ninguno*,*dispositivo_serial=Ninguno*)

Permite el uso del navegador móvil para navegadores que lo admitan.

argumentos

android_activity: el nombre del paquete de Android para iniciar

establecer_preferencia(*nombre: cadena*,*valor: cadena | entero | booleano*)

Establece una preferencia.

to_capabilities() →dictar

Ordena las opciones de Firefox a un*moz:firefoxOpciones*objeto.

CLAVE = 'moz:firefoxOptions'

propiedad binaria:*FirefoxBinario*

Devuelve la instancia de FirefoxBinary.

propiedad ubicación_binaria: str

Devoluciones

La ubicación del binario.

propiedad default_capabilities: dictar

Devuelve las capacidades mínimas necesarias como diccionario.

preferencias de propiedad: dict

Devoluciones

Un dictado de preferencias.

perfil de propiedad:*Perfil de Firefox*

Devoluciones

El perfil de Firefox a utilizar.

7.13 Perfil del controlador web de Firefox

excepciónselenium.webdriver.firefox.firefox_profile.AddonFormatError(***kwargs*)

Bases:Excepción

Excepción para archivos de manifiesto complementarios que no estén bien formados.

claseselenium.webdriver.firefox.firefox_profile.Perfil de Firefox(*directorio_perfil=Ninguno*)

Bases:objeto

__inicio__(*directorio_perfil=Ninguno*)

Inicializa una nueva instancia de un perfil de Firefox.

argumentos

- directorio_perfil: Directorio del perfil que desea utilizar. Si se pasa un directorio, se clonará y el controlador utilizará el directorio clonado cuando se cree una instancia. El valor predeterminado es Ninguno y creará un nuevo directorio cuando se cree el objeto.

agregar_extensión(*extensión = Ninguna*)

establecer_preferencia(*llave*,*valor*)

Establece la preferencia que queremos en el perfil.

actualizar_preferencias()

Escribe las preferencias del usuario deseadas en el disco.

DEFAULT_PREFERENCES = Ninguno

propiedad aceptar_untrusted_certs

propiedad asumir_untrusted_cert_issuer

propiedad codificada: str

Actualiza las preferencias y crea una cadena comprimida codificada en base64 del directorio de perfil.

ruta de propiedad

Obtiene el directorio de perfil que se está utilizando actualmente.

puerto de propiedad

Obtiene el puerto en el que está trabajando WebDriver.

7.14 Firefox WebDriver binario

claseselenium.webdriver.firefox.firefox_binary.FirefoxBinario(***kwargs*)

Bases:objeto

__inicio__(*firefox_path=Ninguno*,*log_file=Ninguno*)

Crea una nueva instancia del binario de Firefox.

argumentos

- firefox_path: ruta al ejecutable de Firefox. De forma predeterminada, se detectará desde las ubicaciones estándar.

- **log_file: un objeto de archivo al que redirigir la salida del proceso de Firefox. Puede ser sys.stdout.**
  Tenga en cuenta que con la ejecución en paralelo la salida no será síncrona. De forma predeterminada, será redirigido a /dev/null.

add_command_line_options(**argumentos*)

matar()

Mata el navegador.

Esto es útil cuando el navegador está bloqueado.

navegador_lanzamiento(*perfil,tiempo de espera = 30*)

Inicia el navegador para el nombre de perfil dado.

Se supone que el perfil ya existe.

cual(*nombref*)

Devuelve la ruta completa buscando la ruta del nombre dado.

NO_FOCUS_LIBRARY_NAME = 'x_ignore_nofocus.so'

7.15 Conexión de la extensión Firefox WebDriver

7.16 Controlador web de Chrome

claseselenium.webdriver.chrome.webdriver.Controlador web(*opciones:*Opciones*|Ninguno = Ninguno,servicio:* Servicio*|Ninguno = Ninguno,keep_alive: bool = Verdadero*)

Bases:Controlador de cromo

Controla ChromeDriver y le permite controlar el navegador.

__inicio__(*opciones:*Opciones*|Ninguno = Ninguno,servicio:*Servicio*|Ninguno = Ninguno,keep_alive: bool = Verdadero*) → Ninguno

Crea una nueva instancia del controlador Chrome. Inicia el servicio y luego crea una nueva instancia del controlador Chrome.

argumentos

- opciones: esto requiere una instancia de ChromeOptions
- servicio: objeto de servicio para manejar el controlador del navegador si necesita pasar detalles adicionales
- keep_alive: si se debe configurar ChromeRemoteConnection para utilizar HTTP keep-alive.

7.17 Opciones del controlador web de Chrome

claseselenium.webdriver.chrome.opciones.Opciones

Bases:Opciones de cromo

habilitar_móvil(*android_package: str = 'com.android.chrome',actividad_android: str | Ninguno = Ninguno, dispositivo_serial: cadena | Ninguno = Ninguno*) → Ninguno

Permite el uso del navegador móvil para navegadores que lo admitan.

argumentos

android_activity: el nombre del paquete de Android para iniciar

propiedad default_capabilities: dict

Devuelve las capacidades mínimas necesarias como diccionario.

7.18 Servicio Chrome WebDriver

claseselenium.webdriver.chrome.servicio.Servicio(*ruta_ejecutable=Ninguno,puerto: int = 0,argumentos_servicio: Lista[cadena] | Ninguno = Ninguno,salida_log: int | cadena | IO[Cualquiera] | Ninguno = Ninguno,env: Mapeo[cadena, cadena] | Ninguno = Ninguno,** kwargs*)

Bases:Servicio de cromo

Una clase de servicio que es responsable del inicio y la parada de*controlador cromado*.

Parámetros

- ruta_ejecutable –ruta de instalación del ejecutable de chromedriver, por defecto es*controlador cromado*.
- puerto -El puerto en el que se ejecutará el servicio tiene por defecto 0, donde el sistema operativo decidirá.
- argumentos_servicio – (Opcional) Lista de argumentos que se pasarán al subproceso al iniciar el ejecutable.
- salida_log – (Opcional) representación int de STDOUT/DEVNULL, cualquier instancia de IO o ruta de cadena al archivo.
- entorno – (Opcional) Mapeo de variables de entorno para el nuevo proceso, el valor predeterminado es *os.environ*.

__inicio__(*ruta_ejecutable=Ninguno,puerto: int = 0,service_args: Lista[cadena] | Ninguno = Ninguno,salida_log: int | cadena | IO[Cualquiera] | Ninguno = Ninguno,env: Mapeo[cadena, cadena] | Ninguno = Ninguno,**kwargs*)→Ninguno

7.19 Controlador web remoto

La implementación de WebDriver.

claseselenium.webdriver.remote.webdriver.Controlador web base

Bases:objeto

Clase base abstracta para todos los subtipos de Webdriver.

ABC permite registrar implementaciones personalizadas de Webdriver para que las comprobaciones del tipo de instancia se realicen correctamente.

claseselenium.webdriver.remote.webdriver.Controlador web(*command_executor='http://127.0.0.1:4444'*, *keep_alive=Verdadero,file_detector=Ninguno,opciones: OpcionesBase | Lista[OpcionesBase] | Ninguno = Ninguno*)

Bases:*Controlador web base*

Controla un navegador enviando comandos a un servidor remoto. Se espera que este servidor ejecute el protocolo de conexión WebDriver tal como se define enhttps://www.selenium.dev/documentation/legacy/json_wire_protocol/.

Atributos

• session_id: ID de cadena de la sesión del navegador iniciada y controlada por este WebDriver.

- **capacidades - Diccionario de capacidades efectivas de esta sesión del navegador tal como se devuelve** por el servidor remoto. Verhttps://www.selenium.dev/documentation/legacy/desired_capacity/

• command_executor: objeto remote_connection.RemoteConnection utilizado para ejecutar comandos.

• error_handler: objeto errorhandler.ErrorHandler utilizado para manejar errores.

__inicio__(*command_executor='http://127.0.0.1:4444'*,*keep_alive=Verdadero*,*file_detector=Ninguno*,*opciones: Opciones Base | Lista[OpcionesBase] | Ninguno = Ninguno*) →Ninguno

Cree un nuevo controlador que emita comandos utilizando el protocolo de conexión.

argumentos

- **command_executor: una cadena que representa la URL del servidor remoto o una costumbre** conexión_remota.Objeto RemoteConnection. El valor predeterminado es 'http://127.0.0.1:4444/wd/centro'.
- **keep_alive: si se debe configurar remote_connection.RemoteConnection para usar** Mantener vivo HTTP. El valor predeterminado es Verdadero.
- **file_detector: pasa el objeto detector de archivos personalizado durante la creación de instancias. Si ninguno,** entonces se utilizará el LocalFileDetector() predeterminado.

• opciones: instancia de un controlador opciones. Clase de opciones

agregar_cookie(*dictado de cookies*) →Ninguno

Agrega una cookie a su sesión actual.

argumentos

- **cookie_dict: un objeto de diccionario, con las claves requeridas: "nombre" y "valor";** claves opcionales: “ruta”, “dominio”, “seguro”, “httpOnly”, “expiry”, “sameSite”

Uso

```
conductor.agregar_cookie({'nombre':'foo','valor':'bar'})
conductor.agregar_cookie({'nombre':'foo','valor':'bar','camino':'/'}) conductor.
agregar_cookie({'nombre':'foo','valor':'bar','camino':'/',
→'seguro':Verdadero})
conductor.agregar_cookie({'nombre':'foo','valor':'bar','mismo sitio':
→'Estricto'})
```

agregar_credencial(*credencial: Credencial*) →Ninguno

Inyecta una credencial en el autenticador.

add_virtual_authenticator(*opciones: Opciones de VirtualAuthenticator*) →Ninguno

Agrega un autenticador virtual con las opciones dadas.

atrás() →Ninguno

Retrocede un paso en el historial del navegador.

Uso

```
conductor.atrás()
```

bidi_conexión()

cerca() →Ninguno

Cierra la ventana actual.

Uso

```
conductor.cerca()
```

crear_web_elemento(*elemento_id: cadena*) →*Elemento web*

Crea un elemento web con el especificado.*id_elemento*.

eliminar_todas_las_cookies() →Ninguno

Elimine todas las cookies en el ámbito de la sesión.

Uso

```
conductor.eliminar_todas_las_cookies()
```

eliminar_cookie(*nombre*) →Ninguno

Elimina una sola cookie con el nombre de pila.

Uso

```
conductor.eliminar_cookie('mi_cookie')
```

eliminar_archivos_descargables() →Ninguno

Elimina todos los archivos descargables.

descargar_archivo(*nombre_archivo: cadena*,*directorio_destino: cadena*) →Ninguno

Descarga un archivo con el nombre de archivo especificado al directorio de destino.

file_name: el nombre del archivo a descargar. target_directory: la ruta al directorio para guardar el archivo descargado.

ejecutar(*comando_controlador: cadena*,*parámetros: dict | Ninguno = Ninguno*) →dictar

Envía un comando para que lo ejecute un comando.CommandExecutor.

argumentos

- driver_command: el nombre del comando a ejecutar como una cadena.
- params: un diccionario de parámetros con nombre para enviar con el comando.

Devoluciones

La respuesta JSON del comando se carga en un objeto de diccionario.

ejecutar_async_script(*guión: cadena*,**argumentos*)

Ejecuta JavaScript de forma asincrónica en la ventana/marco actual.

argumentos

- script: El JavaScript a ejecutar.
- *args: cualquier argumento aplicable a su JavaScript.

Uso

```
guion="var devolución de llamada = argumentos[argumentos.longitud - 1]; "\
      "window.setTimeout(function(){ callback('timeout') }, 3000);" conductor.
      ejecutar_async_script(guión)
```

ejecutar_script(*guion*, **argumentos*)

Ejecuta JavaScript de forma sincrónica en la ventana/marco actual.

argumentos

- script: El JavaScript a ejecutar.
- *args: cualquier argumento aplicable a su JavaScript.

Uso

```
conductor.ejecutar_script('devolver documento.título;')
```

archivo_detector_context(*clase_detector_archivo*, **argumentos*, ***kwargs*)

Anula el detector de archivos actual (si es necesario) en un contexto limitado. Garantiza que el detector de archivos original esté configurado posteriormente.

Ejemplo:

```
concontrolador web.file_detector_context(Detector de archivos inútil):
    alguna entrada.enviar_claves('/etc/hosts')
```

argumentos

- **file_detector_class: clase del detector de archivos deseado. Si la clase es diferente** desde el file_detector actual, luego se crea una instancia de la clase con args y kwargs y se usa como detector de archivos durante la duración del administrador de contexto.
- **args: argumentos opcionales que se pasan a la clase detectora de archivos durante** creación de instancias.
- kwargs: argumentos de palabras clave, pasados de la misma manera que args.

encontrar_elemento(*por = 'identificación'*, *valor: cadena | Ninguno = Ninguno*) → *Elemento web*

Encuentre un elemento dado una estrategia Por y un localizador.

Uso

```
elemento=conductor.encontrar_elemento(Por.IDENTIFICACIÓN,'foo')
```

Tipo de devolución

Elemento web

buscar_elementos(*por = 'identificación'*, *valor: cadena | Ninguno = Ninguno*) → Lista[*Elemento web*]

Encuentra elementos dados por estrategia y localizador.

Uso

```
elementos=conductor.buscar_elementos(Por.CLASE_NOMBRE,'foo')
```

Tipo de devolución

lista de *Elemento web*

adelante() → Ninguno

Va un paso adelante en el historial del navegador.

Uso

```
conductor.adelante()
```

ventana_pantalla completa() →Ninguno

Invoca la operación de 'pantalla completa' específica del administrador de ventanas.

conseguir(*URL: cadena*) →Ninguno

Carga una página web en la sesión actual del navegador.

obtener_cookie(*nombre*) →Dictado | Ninguno

Obtenga una sola cookie por nombre. Devuelve la cookie si se encuentra, Ninguna en caso contrario.

Uso

```
conductor.obtener_cookie('mi_cookie')
```

obtener_cookies() →Lista[dict]

Devuelve un conjunto de diccionarios, correspondientes a las cookies visibles en la sesión actual.

Uso

```
conductor.obtener_cookies()
```

obtener_credenciales() →Lista[Credencial]

Devuelve la lista de credenciales propiedad del autenticador.

get_downloadable_files() →dictar

Recupera los archivos descargables como un mapa de nombres de archivos y sus URL correspondientes.

obtener_log(*tipo_log*)

Obtiene el registro para un tipo de registro determinado.

argumentos

- log_type: tipo de registro que será devuelto

Uso

```
conductor.obtener_log('navegador')
conductor.obtener_log('conductor')
conductor.obtener_log('cliente')
conductor.obtener_log('servidor')
```

get_pinned_scripts() →Lista[cadena]

get_screenshot_as_base64() →cadena

Obtiene la captura de pantalla de la ventana actual como una cadena codificada en base64 que es útil en imágenes incrustadas en HTML.

Uso

```
conductor.get_screenshot_as_base64()
```

get_screenshot_as_file(*Nombre del archivo*) →booleano

Guarda una captura de pantalla de la ventana actual en un archivo de imagen PNG. Devuelve False si hay algún IOError; en caso contrario, devuelve True. Utilice rutas completas en su nombre de archivo.

argumentos

- nombre de archivo: la ruta completa en la que desea guardar su captura de pantalla. Esto debería terminar con un.*png* extensión.

Uso

```
conductor.get_screenshot_as_file('/Capturas de pantalla/foo.png')
```

get_screenshot_as_png() →bytes

Obtiene la captura de pantalla de la ventana actual como datos binarios.

Uso

```
conductor.get_screenshot_as_png()
```

get_window_position(*windowHandle='actual'*) →dictar

Obtiene la posición x,y de la ventana actual.

Uso

```
conductor.get_window_position()
```

get_window_rect() →dictar

Obtiene las coordenadas x, y de la ventana, así como el alto y el ancho de la ventana actual.

Uso

```
conductor.get_window_rect()
```

get_window_size(*manejador de ventana: str = 'actual'*) →dictar

Obtiene el ancho y el alto de la ventana actual.

Uso

```
conductor.get_window_size()
```

implícitamente_esperar(*time_to_wait: flotar*) →Ninguno

Establece un tiempo de espera fijo para esperar implícitamente a que se encuentre un elemento o se complete un comando. Solo es necesario llamar a este método una vez por sesión. Para configurar el tiempo de espera para las llamadas a ejecutar_async_script, consulte set_script_timeout.

argumentos

- time_to_wait: cantidad de tiempo de espera (en segundos)

Uso

```
conductor.implícitamente_esperar(30)
```

maximizar_ventana() →Ninguno

Maximiza la ventana actual que está utilizando webdriver.

minimizar_ventana() →Ninguno

Invoca la operación 'minimizar' específica del administrador de ventanas.

pin_script(*guión: cadena*,*script_key=Ninguno*) →Clave de secuencia de comandos

Almacene scripts de JavaScript comunes para ejecutarlos más tarde mediante una ID única con función hash.

imprimir_página(*opciones_impresión: Opciones de impresión | Ninguno = Ninguno*) →cadena

Toma el PDF de la página actual.

El controlador hace todo lo posible para devolver un PDF según los parámetros proporcionados.

abandonar() →Ninguno

Sale del controlador y cierra todas las ventanas asociadas.

Uso

```
conductor.abandonar()
```

refrescar() →Ninguno

Actualiza la página actual.

Uso

```
conductor.refrescar()
```

eliminar_todas_credenciales() →Ninguno

Elimina todas las credenciales del autenticador.

eliminar_credencial(*credencial_id: cadena | matriz de bytes*) →Ninguno

Elimina una credencial del autenticador.

eliminar_autenticador_virtual() →Ninguno

Elimina un autenticador virtual agregado previamente.

El autenticador ya no es válido después de su eliminación, por lo que no se puede llamar a ningún método.

guardar_captura de pantalla (*Nombre del archivo*) →booleano

Guarda una captura de pantalla de la ventana actual en un archivo de imagen PNG. Devuelve False si hay algún IOError; en caso contrario, devuelve True. Utilice rutas completas en su nombre de archivo.

argumentos

- nombre de archivo: la ruta completa en la que desea guardar su captura de pantalla. Esto debería terminar con un.*png* extensión.

Uso

```
conductor.guardar_captura de pantalla ('/Capturas de pantalla/foo.png')
```

set_page_load_timeout(*time_to_wait: flotar*) →Ninguno

Establezca la cantidad de tiempo que se debe esperar a que se complete la carga de una página antes de generar un error.

argumentos

- time_to_wait: la cantidad de tiempo de espera

Uso

```
conductor.set_page_load_timeout(30)
```

set_script_timeout(*time_to_wait: flotar*) →Ninguno

Establezca la cantidad de tiempo que el script debe esperar durante una llamada a ejecutar_async_script antes de generar un error.

argumentos

- time_to_wait: la cantidad de tiempo de espera (en segundos)

Uso

```
conductor.set_script_timeout(30)
```

set_user_verified(*verificado: booleano*) →Ninguno

Establece si el autenticador simulará el éxito o fallará en la verificación del usuario.

verificado: Verdadero si el autenticador pasará la verificación del usuario; Falso en caso contrario.

establecer_posición_ventana(*incógnita*,*y*,*manejador de ventana: str = 'actual'*) →dictar

Establece la posición x,y de la ventana actual. (ventana.moveTo)

argumentos

- x: la coordenada x en píxeles para establecer la posición de la ventana
- y: la coordenada y en píxeles para establecer la posición de la ventana

Uso

```
conductor.establecer_posición_ventana(0,0)
```

set_window_rect(*x=Ninguno*,*y=Ninguno*,*ancho=Ninguno*,*altura=Ninguno*) →dictar

Establece las coordenadas x, y de la ventana, así como la altura y el ancho de la ventana actual. Este método sólo es compatible con navegadores compatibles con W3C; otros navegadores deberían usar*establecer_posición_ventana*y *establecer_tamaño_ventana*.

Uso

```
conductor.set_window_rect(x=10, y=10)
conductor.set_window_rect(ancho=100, altura=200)
conductor.set_window_rect(x=10, y=10, ancho=100, altura=200)
```

set_window_size(*ancho*,*altura*,*manejador de ventana: str = 'actual'*) →Ninguno

Establece el ancho y alto de la ventana actual. (ventana.resizeTo)

argumentos

- ancho: el ancho en píxeles para configurar la ventana
- altura: la altura en píxeles para configurar la ventana

Uso

```
conductor.set_window_size(800,600)
```

inicio_cliente()

Llamado antes de iniciar una nueva sesión.

Este método puede anularse para definir un comportamiento de inicio personalizado.

inicio_sesión(*capacidades: dict*) →Ninguno

Crea una nueva sesión con las capacidades deseadas.

argumentos

- capacidades: un dictado de capacidades con el que iniciar la sesión.

detener_cliente()

Se llama después de ejecutar un comando de salida.

Este método puede anularse para definir un comportamiento de apagado personalizado.

desprender(*script_key: clave de script*) →Ninguno

Eliminar un script anclado del almacenamiento.

capacidades de propiedad: dict

Devuelve las capacidades actuales del controlador que se están utilizando.

propiedad URL_actual: str

Obtiene la URL de la página actual.

Uso

```
conductor.URL_actual
```

propiedad current_window_handle: str

Devuelve el identificador de la ventana actual.

Uso

```
conductor.identificador_ventana_actual
```

propiedad file_detector: FileDetector

propiedad log_types

Obtiene una lista de los tipos de registros disponibles. Esto sólo funciona con navegadores compatibles con w3c.

Uso

```
conductor.tipos de registro
```

propiedad móvil: *Móvil*

nombre de la propiedad: cadena

Devuelve el nombre del navegador subyacente para esta instancia.

Uso

```
nombre=conductor.nombre
```

orientación de la propiedad

Obtiene la orientación actual del dispositivo.

Uso

```
orientación=conductor.orientación
```

propiedad page_source: str

Obtiene el origen de la página actual.

Uso

```
conductor.fuente_página
```

propiedad switch_to: CambiarA

Devoluciones

- SwitchTo: un objeto que contiene todas las opciones para cambiar el foco.

Uso

```
elemento=conductor.cambiar_a.alerta de
elemento_activo=conductor.cambiar_a.conductor
alerta.cambiar_a.controlador default_content().
cambiar_a.marco('nombre_marco') conductor.
cambiar_a.marco(1)
conductor.cambiar_a.marco (conductor.buscar_elementos(Por.ETIQUETA_NOMBRE,"iframe")[0])
conductor.cambiar_a.controlador parent_frame().cambiar_a.ventana('principal')
```

tiempos de espera de propiedad: Tiempos de espera

Obtenga todos los tiempos de espera establecidos en la sesión actual.

Uso

```
conductor.tiempos de espera
```

Tipo de devolución

Se acabó el tiempo

título de propiedad: str

Devuelve el título de la página actual.

Uso

```
título=conductor.título
```

propiedad virtual_authenticator_id: str

Devuelve la identificación del autenticador virtual.

propiedad window_handles: Lista[cadena]

Devuelve los identificadores de todas las ventanas dentro de la sesión actual.

Uso

```
conductor.manijas_ventana
```

selenium.webdriver.remote.webdriver.crear_coincidencias(*opciones: Lista[BaseOptions]*) →dictar

selenium.webdriver.remote.webdriver.get_remote_connection(*capacidades*,*comando_ejecutor*, *mantener_alive*,*ignore_local_proxy=Falso*)

selenium.webdriver.remote.webdriver.importar_cdp()

7.20 Elemento web del controlador web remoto

claseselenium.webdriver.remote.webelement.Elemento WebBase

Bases:objeto

Clase base abstracta para WebElement.

ABC permitirá registrar tipos personalizados como WebElement para pasar las comprobaciones de tipo.

claseselenium.webdriver.remote.webelement.Elemento Web(*padre*,*identificación_*)

Bases:*Elemento WebBase*

Representa un elemento DOM.

Generalmente, todas las operaciones interesantes que interactúan con un documento se realizarán a través de esta interfaz.

Todas las llamadas a métodos realizarán una verificación de actualización para garantizar que la referencia del elemento siga siendo válida. esencialmente determina si el elemento todavía está adjunto al DOM. Si esta prueba falla, entonces un Excepción de referencia de elemento obsoletose lanza y todas las llamadas futuras a esta instancia fallarán.Este

__inicio__(*padre*,*identificación_*) →Ninguno

claro() →Ninguno

Borra el texto si es un elemento de entrada de texto.

hacer clic() →Ninguno

Hace clic en el elemento.

encontrar_elemento(*por = 'identificación'*,*valor=Ninguno*) →*Elemento web*

Encuentre un elemento dado una estrategia Por y un localizador.

Uso

```
elemento=elemento.encontrar_elemento(Por.IDENTIFICACIÓN,'foo')
```

Tipo de devolución

Elemento web

buscar_elementos(*por = 'identificación'*,*valor=Ninguno*) →Lista[*Elemento web*]

Encuentra elementos dados por estrategia y localizador.

Uso

```
elemento=elemento.buscar_elementos(Por.CLASE_NOMBRE,'foo')
```

Tipo de devolución

lista de*Elemento web*

obtener_atributo(*nombre*) →cadena | Ninguno

Obtiene el atributo o propiedad dado del elemento.

Este método primero intentará devolver el valor de una propiedad con el nombre de pila. Si una propiedad con ese nombre no existe, devuelve el valor del atributo con el mismo nombre. Si no hay ningún atributo con ese nombre, Ningunoes devuelto.

Los valores que se consideran verdaderos, es decir, "verdaderos" o "falso", se devuelven como booleanos. Todos los demás no-Ningunolos valores se devuelven como cadenas. Para atributos o propiedades que no existen,Ningunoes devuelto.

Para obtener el valor exacto del atributo o propiedad, utiliceget_dom_attribute()oobtener_propiedad() métodos respectivamente.

argumentos

- nombre: nombre del atributo/propiedad a recuperar.

Ejemplo:

```
# Comprobar si la clase CSS "activa" se aplica a un elemento. está_activo
="activo"enelemento_objetivo.obtener_atributo("clase")
```

get_dom_attribute(*nombre*) →cadena

Obtiene el atributo dado del elemento. A diferencia deget_atributo(),este método solo devuelve atributos declarados en el marcado HTML del elemento.

argumentos

- nombre: nombre del atributo a recuperar.

Uso

```
longitud_texto=elemento_objetivo.get_dom_attribute("clase")
```

obtener_propiedad(*nombre*) →cadena | booleano | *Elemento web* | dictar

Obtiene la propiedad dada del elemento.

argumentos

- nombre: nombre de la propiedad a recuperar.

Uso

```
longitud_texto=elemento_objetivo.obtener_propiedad("longitud_texto")
```

está_displayed() →booleano

Si el elemento es visible para un usuario.

está_enabled() →booleano

Devuelve si el elemento está habilitado.

está_seleccionado() →booleano

Devuelve si el elemento está seleccionado.

Se puede utilizar para comprobar si una casilla de verificación o un botón de opción está seleccionado.

captura de pantalla (*Nombre del archivo*) →booleano

Guarda una captura de pantalla del elemento actual en un archivo de imagen PNG. Devuelve False si hay algún IOError; en caso contrario, devuelve True. Utilice rutas completas en su nombre de archivo.

argumentos

- nombre de archivo: la ruta completa en la que desea guardar su captura de pantalla. Esto debería terminar con un.*png* extensión.

Uso

```
elemento.captura de pantalla ('/Capturas de pantalla/foo.png')
```

enviar_claves(**valor: cadena*) →Ninguno

Simula escribir en el elemento.

argumentos

- valor: una cadena para escribir o configurar campos de formulario. Para configurar entradas de archivos, esta podría ser una ruta de archivo local.

Utilice esto para enviar eventos clave simples o para completar campos de formulario:

```
campo_texto_formulario=conductor.encontrar_elemento(Por.NOMBRE,'nombre de usuario')
campo_texto_formulario.enviar_claves("administración")
```

Esto también se puede utilizar para configurar entradas de archivos.

```
entrada_archivo=conductor.encontrar_elemento(Por.NOMBRE,'foto de perfil'
) entrada_archivo.enviar_claves("ruta/a/imagendeperfil.gif")
# Generalmente es mejor ajustar la ruta del archivo en uno de los métodos
# en os.path para devolver la ruta real para admitir pruebas entre sistemas operativos.
# file_input.send_keys(os.path.abspath("ruta/a/profilepic.gif"))
```

entregar() →Ninguno

Envía un formulario.

valor_de_css_property(*nombre_propiedad*) →cadena

El valor de una propiedad CSS.

propiedad nombre_accesible: cadena

Devuelve el nivel ARIA del elemento web actual.

propiedad aria_role: str

Devuelve el rol ARIA del elemento web actual.

ID de propiedad: cadena

ID interna utilizada por selenium.

Esto es principalmente para uso interno. Los casos de uso simples, como verificar si 2 elementos web se refieren al mismo elemento, se pueden realizar usando ==:

```
sielemento1==elemento2:
    imprimir("Estos 2 son iguales")
```

ubicación de la propiedad: dict

La ubicación del elemento en el lienzo renderizable.

propiedad location_once_scrolled_into_view: dict

ESTA PROPIEDAD PUEDE CAMBIAR SIN PREVIO AVISO. Utilice esto para descubrir en qué parte de la pantalla se encuentra un elemento para que podamos hacer clic en él. Este método debería hacer que el elemento se desplace a la vista.

Devuelve la ubicación de la esquina superior izquierda de la pantalla, o coordenadas cero si el elemento no es visible.

padre de propiedad

Referencia interna a la instancia de WebDriver desde la que se encontró este elemento.

propiedad recta: dict

Un diccionario con el tamaño y ubicación del elemento.

propiedad capture_as_base64: str

Obtiene la captura de pantalla del elemento actual como una cadena codificada en base64.

Uso

```
img_b64=elemento.captura de pantalla_as_base64
```

propiedad capture_as_png: bytes

Obtiene la captura de pantalla del elemento actual como datos binarios.

Uso

```
elemento_png=elemento.captura de pantalla_como_png
```

propiedad shadow_root: ShadowRoot

Devuelve una raíz oculta del elemento si hay una o un error. Sólo funciona desde Chromium 96, Firefox 96 y Safari 16.4 en adelante.

Devoluciones

- Objeto ShadowRoot o
- NoSuchShadowRoot: si no se adjuntó ninguna raíz de sombra al elemento

tamaño de propiedad: dict

El tamaño del elemento.

propiedad nombre_etiqueta: str

este elementonombre de etiquetapropiedad.

texto de propiedad: str

El texto del elemento.

7.21 Comando remoto de WebDriver

claseselenium.webdriver.remote.command.Dominio

Bases:objeto

Define constantes para los comandos estándar de WebDriver.

Si bien estas constantes no tienen significado en sí mismas, se utilizan para ordenar comandos a través de un servicio que implementa el protocolo de conexión remota de WebDriver:

https://w3c.github.io/webdriver/

ADD_COOKIE: str = 'añadirCookie'

ADD_CREDENTIAL: str = 'añadirCredencial'

ADD_VIRTUAL_AUTHENTICATOR: str = 'addVirtualAuthenticator'

CLEAR_ELEMENT: str = 'clearElement'

CLICK_ELEMENT: str = 'clickElement'

CERRAR: str = 'cerrar'

CONTEXT_HANDLES: str = 'getContextHandles'

CURRENT_CONTEXT_HANDLE: str = 'getCurrentContextHandle'

DELETE_ALL_COOKIES: str = 'eliminarTodas las Cookies'

DELETE_COOKIE: str = 'eliminarCookie'

DELETE_DOWNLOADABLE_FILES: str = 'eliminar archivos descargables'

DELETE_SESSION: str = 'eliminar sesión'

DOWNLOAD_FILE: str = 'descargar archivo'

ELEMENT_SCREENSHOT: str = 'elementoCaptura de pantalla'

EXECUTE_ASYNC_SCRIPT: str = 'ejecutarAsyncScript'

FIND_CHILD_ELEMENT: str = 'encontrarChildElement'

FIND_CHILD_ELEMENTS: str = 'buscarElementosChild'

FIND_ELEMENT: str = 'buscarElemento'

FIND_ELEMENTS: str = 'buscarElementos'

FIND_ELEMENTS_FROM_SHADOW_ROOT: str = 'findElementsFromShadowRoot'

FIND_ELEMENT_FROM_SHADOW_ROOT: str = 'findElementFromShadowRoot'

FULLSCREEN_WINDOW: str = 'ventana de pantalla completa'

OBTENER: cadena = 'obtener'

GET_ALL_COOKIES: str = 'obtenerCookies'

GET_AVAILABLE_LOG_TYPES: str = 'getAvailableLogTypes'

GET_COOKIE: str = 'obtenerCookie'

GET_CREDENTIALS: str = 'obtenerCredenciales'

GET_CURRENT_URL: cadena = 'getCurrentUrl'

GET_DOWNLOADABLE_FILES: str = 'getDownloadableFiles'

GET_ELEMENT_ARIA_LABEL: str = 'getElementAriaLabel'

GET_ELEMENT_ARIA_ROLE: str = 'getElementAriaRole'

GET_ELEMENT_ATTRIBUTE: str = 'getElementAttribute'

GET_ELEMENT_PROPERTY: str = 'getElementProperty'

GET_ELEMENT_RECT: str = 'getElementRect'

GET_ELEMENT_TAG_NAME: str = 'getElementTagName'

GET_ELEMENT_TEXT: str = 'getElementText'

GET_ELEMENT_VALUE_OF_CSS_PROPERTY: str = 'getElementValueOfCssProperty'

GET_LOG: cadena = 'getLog'

GET_NETWORK_CONNECTION: str = 'getNetworkConnection'

GET_PAGE_SOURCE: str = 'getPageSource'

GET_SCREEN_ORIENTATION: str = 'getScreenOrientation'

GET_SHADOW_ROOT: str = 'getShadowRoot'

GET_TIMEOUTS: str = 'getTimeouts'

GET_TITLE: str = 'obtenerTítulo'

GET_WINDOW_RECT: str = 'getWindowRect'

GO_BACK: str = 'volver'

GO_FORWARD: str = 'ir hacia adelante'

IS_ELEMENT_ENABLED: str = 'isElementEnabled'

IS_ELEMENT_SELECTED: str = 'isElementSelected'

MINIMIZE_WINDOW: str = 'minimizarVentana'

NUEVA_SESSIÓN: str = 'nuevaSesión'

NUEVA_VENTANA: str = 'nuevaVentana'

PRINT_PAGE: cadena = 'imprimirPágina'

SALIR: str = 'salir'

ACTUALIZAR: str = 'actualizar'

REMOVE_ALL_CREDENTIALS: str = 'removeAllCredentials'

REMOVE_CREDENTIAL: str = 'eliminarCredencial'

REMOVE_VIRTUAL_AUTHENTICATOR: str = 'eliminarVirtualAuthenticator'

CAPTURA DE PANTALLA: str = 'captura de pantalla'

SEND_KEYS_TO_ELEMENT: str = 'enviarKeysToElement'

SET_NETWORK_CONNECTION: str = 'setNetworkConnection'

SET_SCREEN_ORIENTATION: str = 'setScreenOrientation'

SET_TIMEOUTS: str = 'setTimeouts'

SET_USER_VERIFIED: str = 'setUserVerified'

SET_WINDOW_RECT: str = 'setWindowRect'

SWITCH_TO_CONTEXT: str = 'cambiarAContexto'

SWITCH_TO_FRAME: str = 'cambiar a marco'

SWITCH_TO_PARENT_FRAME: str = 'switchToParentFrame'

SWITCH_TO_WINDOW: str = 'cambiar a ventana'

UPLOAD_FILE: str = 'cargar archivo'

W3C_ACCEPT_ALERT: cadena = 'w3cAcceptAlert'

W3C_ACTIONS: str = 'acciones'

W3C_CLEAR_ACTIONS: str = 'clearActionState'

W3C_DISMISS_ALERT: str = 'w3cDismissAlert'

W3C_EXECUTE_SCRIPT: cadena = 'w3cExecuteScript'

W3C_EXECUTE_SCRIPT_ASYNC: str = 'w3cExecuteScriptAsync'

W3C_GET_ACTIVE_ELEMENT: str = 'w3cGetActiveElement'

W3C_GET_ALERT_TEXT: cadena = 'w3cGetAlertText'

W3C_GET_CURRENT_WINDOW_HANDLE: str = 'w3cGetCurrentWindowHandle'

W3C_GET_WINDOW_HANDLES: str = 'w3cGetWindowHandles'

W3C_MAXIMIZE_WINDOW: str = 'w3cMaximizeWindow'

W3C_SET_ALERT_VALUE: cadena = 'w3cSetAlertValue'

7.22 Controlador de errores del controlador web remoto

claseselenium.webdriver.remote.errorhandler.Código de error

Bases:objeto

Códigos de error definidos en el protocolo de conexión WebDriver.

ELEMENT_CLICK_INTERCEPTED = [64, 'clic en elemento interceptado']

ELEMENT_IS_NOT_SELECTABLE = [15, 'elemento no seleccionable']

ELEMENT_NOT_INTERACTABLE = [60, 'elemento no interactuable']

ELEMENT_NOT_VISIBLE = [11, 'elemento no visible']

IME_ENGINE_ACTIVATION_FAILED = [31, 'falló la activación del motor de tiempo']

IME_NOT_AVAILABLE = [30, 'tiempo no disponible']

INSECURE_CERTIFICATE = ['certificado inseguro']

INVALID_ARGUMENT = [61, 'argumento no válido']

INVALID_COOKIE_DOMAIN = [24, 'dominio de cookies no válido']

INVALID_COORDINATES = ['coordenadas no válidas']

INVALID_ELEMENT_COORDINATES = [29, 'coordenadas de elemento no válidas']

INVALID_ELEMENT_STATE = [12, 'estado de elemento no válido']

INVALID_SELECTOR = [32, 'selector no válido']

INVALID_SESSION_ID = ['ID de sesión no válida']

INVALID_XPATH_SELECTOR = [51, 'selector no válido']

INVALID_XPATH_SELECTOR_RETURN_TYPER = [52, 'selector no válido']

JAVASCRIPT_ERROR = [17, 'error de javascript']

METHOD_NOT_ALLOWED = [405, 'operación no admitida']

MOVE_TARGET_OUT_OF_BOUNDS = [34, 'mover el objetivo fuera de los límites']

NO_ALERT_OPEN = [27, 'no existe tal alerta']

NO_SUCH_COOKIE = [62, 'no existe tal cookie']

NO_SUCH_ELEMENT = [7, 'no existe tal elemento']

NO_SUCH_FRAME = [8, 'no existe tal marco']

NO_SUCH_SHADOW_ROOT = ['no existe tal raíz de sombra']

NO_SUCH_WINDOW = [23, 'no existe esa ventana']

SCRIPT_TIMEOUT = [28, 'tiempo de espera del script']

SESSION_NOT_CREATED = [33, 'sesión no creada']

STALE_ELEMENT_REFERENCE = [10, 'referencia de elemento obsoleto']

ÉXITO = 0

TIEMPO DE ESPERA = [21, 'tiempo de espera']

UNABLE_TO_CAPTURE_SCREEN = [63, 'no se puede capturar la pantalla']

UNABLE_TO_SET_COOKIE = [25, 'no se puede configurar la cookie']

UNEXPECTED_ALERT_OPEN = [26, 'alerta inesperada abierta']

UNKNOWN_COMMAND = [9, 'comando desconocido']

UNKNOWN_ERROR = [13, 'error desconocido']

UNKNOWN_METHOD = ['excepción de método desconocido']

XPATH_LOOKUP_ERROR = [19, 'selector no válido']

claseselenium.webdriver.remote.errorhandler.Manejador de errores

Bases:objeto

Maneja los errores devueltos por el servidor WebDriver.

respuesta_comprobación(*respuesta: Dict[cadena, Cualquiera]*) →Ninguno

Comprueba que una respuesta JSON del WebDriver no tenga error.

argumentos

- respuesta: la respuesta JSON del servidor WebDriver como un objeto de diccionario.

aumentos

Si la respuesta contiene un mensaje de error.

claseselenium.webdriver.remote.errorhandler.Mapeo de excepciones

Bases:objeto

: Asigna cada código de error en el objeto ErrorCode a la excepción correspondiente. Consultehttps://www.w3.org/TR/webdriver2/#errorspara la especificación w3c

ELEMENT_CLICK_INTERCEPTED

alias de*ElementClickInterceptedException*

ELEMENT_IS_NOT_SELECTABLE

alias de*ElementoNotSelectableException*

ELEMENT_NOT_INTERACTABLE
: alias de*ElementoNotInteractableException*

ELEMENT_NOT_VISIBLE
: alias de*ElementoNotVisibleException*

IME_ENGINE_ACTIVATION_FAILED
: alias de*ImeActivationFailedException*

IME_NOT_AVAILABLE
: alias de*Excepción ImeNotAvailable*

CERTIFICADO_INSEGURO
: alias de*Excepción de certificado inseguro*

ARGUMENTO_INVÁLIDO
: alias de*Excepción de argumento no válido*

DOMINIO_COOKIE_INVALID
: alias de*Excepción de dominio de cookie no válida*

COORDENADAS_INVÁLIDAS
: alias de*Excepción de coordenadas no válidas*

INVALID_ELEMENT_STATE
: alias de*Excepción de estado de elemento no válido*

SELECTOR_INVÁLIDO
: alias de*Excepción de selección no válida*

INVALID_SESSION_ID
: alias de*Excepción de ID de sesión no válida*

INVALID_XPATH_SELECTOR
: alias de*Excepción de selección no válida*

INVALID_XPATH_SELECTOR_RETURN_TYPER
: alias de*Excepción de selección no válida*

JAVASCRIPT_ERROR
: alias de*Excepción Javascript*

MOVE_TARGET_OUT_OF_BOUNDS
: alias de*Excepción MoveTargetOutOfBounds*

NO_ALERT_OPEN
: alias de*NoAlertPresentException*

NO_TALES_COOKIE
: alias de*Ninguna excepción de cookie*

NO_SUCH_ELEMENT
: alias de*Ninguna excepción de elemento tal*

NO_TAL_FRAME
: alias de*Ninguna excepción de marco*

NO_SUCH_SHADOW_ROOT

alias de*NoSuchShadowRootException*

NO_TAL_VENTANA

alias de*Ninguna excepción de ventana tal*

SCRIPT_TIMEOUT

alias de*Excepción de tiempo de espera*

SESIÓN_NO_CREADA

alias de*Excepción de sesión no creada*

STALE_ELEMENT_REFERENCE

alias de*Excepción de referencia de elemento obsoleto*

SE ACABÓ EL TIEMPO

alias de*Excepción de tiempo de espera*

UNABLE_TO_CAPTURE_SCREEN

alias de*Excepción de captura de pantalla*

UNABLE_TO_SET_COOKIE

alias de*No se puede establecer una excepción de cookie*

INESPECTED_ALERT_OPEN

alias de*Excepción de presentación de alerta inesperada*

ERROR_DESCONOCIDO

alias de*Excepción del controlador web*

MÉTODO_DESCONOCIDO

alias de*Excepción de método desconocido*

7.23 WebDriver remoto móvil

claseselenium.webdriver.remote.mobile.Móvil(*conductor*)

Bases:objeto

Tipo de conexión

alias de _Tipo de conexión

__inicio__(*conductor*)

set_network_connection(*red*)

Configure la conexión de red para el dispositivo remoto.

Ejemplo de configuración del modo avión:

```
conductor.móvil.set_network_connection(controlador.móvil.MODO_AVIÓN)
```

AIRPLANE_MODE = <objeto selenium.webdriver.remote.mobile._ConnectionType>

ALL_NETWORK = <objeto selenium.webdriver.remote.mobile._ConnectionType>

DATA_NETWORK = <objeto selenium.webdriver.remote.mobile._ConnectionType>

WIFI_NETWORK = <objeto selenium.webdriver.remote.mobile._ConnectionType>

contexto de propiedad

Devuelve el contexto actual (Nativo o WebView).

contextos de propiedad

Devuelve una lista de contextos disponibles.

propiedad conexión_red

7.24 Conexión remota de WebDriver

claseselenium.webdriver.remote.remote_connection.Conexión remota(*dirección_servidor_remoto: cadena*, *keep_alive: bool = Falso*, *ignore_proxy: bool = Falso*)

Bases:objeto

Una conexión con el servidor WebDriver remoto.

Se comunica con el servidor mediante el protocolo de conexión WebDriver:https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol

__inicio__(*dirección_servidor_remoto: cadena*,*keep_alive: bool = Falso*,*ignore_proxy: bool = Falso*)

cerca()

Limpie los recursos cuando termine con la conexión_remota.

ejecutar(*dominio*,*parámetros*)

Envíe un comando al servidor remoto.

Cualquier sustitución de ruta requerida para la URL asignada al comando debe incluirse en los parámetros del comando.

argumentos

- comando: una cadena que especifica el comando a ejecutar.
- params: un diccionario de parámetros con nombre para enviar con el comando como carga útil JSON.

método de clase get_certificate_bundle_path()

Devoluciones

Rutas del certificado codificado .pem para verificar la conexión con el ejecutor de comandos. El valor predeterminado es certifi.where() o la variable de entorno REQUESTS_CA_BUNDLE si está configurada.

método de clase get_remote_connection_headers(*URL analizada*,*keep_alive=Falso*)

Obtenga encabezados para solicitudes remotas.

argumentos

- parsed_url: la URL analizada
- keep_alive (booleano): ¿es esta una conexión keep-alive (valor predeterminado: Falso)?

método de clase get_timeout()

Devoluciones

Valor de tiempo de espera en segundos para todas las solicitudes http realizadas a la conexión remota

método de clase reset_timeout()

Restablezca el tiempo de espera de la solicitud http en socket._GLOBAL_DEFAULT_TIMEOUT.

método de clase set_certificate_bundle_path(*camino*)

Establezca la ruta al paquete de certificados para verificar la conexión con el ejecutor de comandos. También se puede configurar en Ninguno para deshabilitar la validación de certificados.

argumentos

• ruta: ruta de una cadena de certificados codificada en .pem.

método de clase set_timeout(*se acabó el tiempo*)

Anule el tiempo de espera predeterminado.

argumentos

• timeout: valor de tiempo de espera para solicitudes http en segundos

nombre_navegador = Ninguno

7.25 Utilidades remotas de WebDriver

selenium.webdriver.remote.utils.volcado_json(*json_struct: cualquiera*) →cadena

selenium.webdriver.remote.utils.cargar_json(*s: cadena | bytes*) →Cualquier

7.26 Controlador web de Internet Explorer

claseselenium.webdriver.es.webdriver.Controlador web(*opciones: Opciones | Ninguno = Ninguno,servicio: Servicio | Ninguno = Ninguno,keep_alive: bool = Verdadero*)

Bases:*Controlador web*

Controla IEServerDriver y le permite manejar Internet Explorer.

__inicio__(*opciones: Opciones | Ninguno = Ninguno,servicio: Servicio | Ninguno = Ninguno,keep_alive: bool = Verdadero*) → Ninguno

Crea una nueva instancia del controlador IE.

Inicia el servicio y luego crea una nueva instancia del controlador IE.

argumentos

• opciones: instancia de Opciones de IE, que proporciona opciones de IE adicionales

• servicio: instancia de servicio (opcional) para gestionar el arranque y la parada del conductor.

• keep_alive: si se debe configurar RemoteConnection para utilizar HTTP keep-alive.

abandonar() →Ninguno

Cierra el navegador y cierra el ejecutable de IEServerDriver.

7.27 Controlador web Safari

claseselenium.webdriver.safari.webdriver.Controlador web(*keep_alive=Verdadero*,*opciones: Opciones | Ninguno = Ninguno*,*servicio:* Servicio *| Ninguno = Ninguno*)

Bases: *Controlador web*

Controla SafariDriver y le permite controlar el navegador.

__inicio__(*keep_alive=Verdadero*,*opciones: Opciones | Ninguno = Ninguno*,*servicio:* Servicio *| Ninguno = Ninguno*) → Ninguno

Crea una nueva instancia del controlador Safari e inicia o encuentra un servicio safaridriver en ejecución.

argumentos

- **keep_alive: si se debe configurar SafariRemoteConnection para usar** Mantener vivo HTTP. El valor predeterminado es Verdadero.
- opciones: instancia deopciones.Opciones.
- servicio: objeto de servicio para manejar el controlador del navegador si necesita pasar detalles adicionales

depurar()

obtener_permiso(*permiso*)

abandonar()

Cierra el navegador y apaga el ejecutable de SafariDriver.

establecer_permiso(*permiso*,*valor*)

7.28 Servicio Safari WebDriver

claseselenium.webdriver.safari.servicio.Servicio(*ruta_ejecutable: str | Ninguno = Ninguno*,*puerto: int = 0*, *service_args: Lista[cadena] | Ninguno = Ninguno*,*env: Mapeo[cadena, cadena] | Ninguno = Ninguno*, *reuse_service=Falso*,***kwargs*)

Bases: *Servicio*

Una clase de servicio que es responsable del inicio y la parada de*conductor de safari*.Esto sólo es compatible con MAC OSX.

Parámetros

- ruta_ejecutable –ruta de instalación del ejecutable safaridriver, por defecto */usr/bin/safaridriver*.
- puerto -El puerto en el que se ejecutará el servicio tiene por defecto 0, donde el sistema operativo decidirá.
- argumentos_servicio – (Opcional) Lista de argumentos que se pasarán al subproceso al iniciar el ejecutable.
- entorno – (Opcional) Mapeo de variables de entorno para el nuevo proceso, el valor predeterminado es *os.environ*.

__inicio__(*ruta_ejecutable: str | Ninguno = Ninguno*,*puerto: int = 0*,*service_args: Lista[cadena] | Ninguno = Ninguno*,*entorno: Mapeo[cadena, cadena] | Ninguno = Ninguno*,*reuse_service=Falso*,***kwargs*) → Ninguno

línea_comando_args() →Lista[cadena]

Una lista de argumentos del programa (excluyendo el ejecutable).

propiedad reutilización_servicio: booleano

propiedad service_url: cadena

Obtiene la URL del servicio SafariDriver.

7.29 Seleccionar soporte

claseselenium.webdriver.support.select.Seleccionar(*elemento web:* Elemento web)

Bases:objeto

__inicio__(*elemento web:* Elemento web) →Ninguno

Constructor. Se comprueba que el elemento dado sea, efectivamente, una etiqueta SELECT. Si no es así, se genera una excepción UnexpectedTagNameException.

argumentos

• webelement - SELECCIONAR elemento para envolver

Ejemplo:

desde selenium.webdriver.support.ui importar Seleccionar

Seleccionar(driver.find_element(By.TAG_NAME, “seleccionar”)).select_by_index(2)

deseleccionar_todo() →Ninguno

Borrar todas las entradas seleccionadas.

Esto solo es válido cuando SELECT admite múltiples selecciones. arroja NotImplementedError si SELECT no admite selecciones múltiples

deseleccionar_por_índice(*índice: int*) →Ninguno

Deseleccione la opción en el índice dado. Esto se hace examinando el atributo "índice" de un elemento, y no simplemente contando.

argumentos

• índice: se deseleccionará la opción en este índice.

lanza NoSuchElementException si no hay ninguna opción con el índice especificado en SELECT

deseleccionar_por_valor(*valor: cadena*) →Ninguno

Deseleccione todas las opciones que tengan un valor que coincida con el argumento. Es decir, cuando se le da "foo", esto anularía la selección de una opción como:

<option value=”foo”>Barra</option>

argumentos

• valor: el valor con el que comparar

lanza NoSuchElementException si no hay ninguna opción con el valor especificado en SELECT

deseleccionar_por_texto_visible(*texto: cadena*) →Ninguno

Anule la selección de todas las opciones que muestren texto que coincida con el argumento. Es decir, cuando se le da "Barra", se anularía la selección de una opción como:

<option value="foo">Barra</option>

argumentos

- texto: el texto visible con el que comparar

seleccionar_por_índice(*índice: int*) →Ninguno

Seleccione la opción en el índice dado. Esto se hace examinando el atributo "índice" de un elemento, y no simplemente contando.

argumentos

- índice: se seleccionará la opción en este índice

lanza NoSuchElementException si no hay ninguna opción con el índice especificado en SELECT

seleccionar_por_valor(*valor: cadena*) →Ninguno

Seleccione todas las opciones que tengan un valor que coincida con el argumento. Es decir, cuando se le da "foo", se seleccionará una opción como:

<option value="foo">Barra</option>

argumentos

- valor: el valor con el que comparar

lanza NoSuchElementException si no hay ninguna opción con el valor especificado en SELECT

seleccionar_por_texto_visible(*texto: cadena*) →Ninguno

Seleccione todas las opciones que muestren texto que coincida con el argumento. Es decir, cuando se le da "Barra", se seleccionará una opción como:

<option value="foo">Barra</option>

argumentos

- texto: el texto visible con el que comparar

lanza NoSuchElementException si no hay ninguna opción con el texto especificado en SELECT

propiedad all_selected_options: Lista[*Elemento web*]

Devuelve una lista de todas las opciones seleccionadas que pertenecen a esta etiqueta de selección.

propiedad primera_opción_seleccionada:*Elemento web*

La primera opción seleccionada en esta etiqueta de selección (o la opción seleccionada actualmente en una selección normal)

opciones de propiedad: Lista[*Elemento web*]

Devuelve una lista de todas las opciones que pertenecen a esta etiqueta de selección.

7.30 Esperar soporte

claseselenium.webdriver.support.esperar.WebDriverEsperar(*conductor: D,tiempo de espera: flotar,frecuencia_encuesta: flotante = 0,5,excepciones_ignoradas: Iterable[Tipo[Excepción]] | Ninguno = Ninguno*)

Bases:Genérico[D]

__inicio__(*conductor: D,tiempo de espera: flotar,frecuencia_encuesta: flotante = 0,5,excepciones_ignoradas: Iterable[Tipo[Excepción]] | Ninguno = Ninguno*)

Constructor, toma una instancia de WebDriver y se agota en segundos.

argumentos

- controlador: instancia de WebDriver (es decir, Firefox, Chrome o Remote) o un WebElement
- tiempo de espera: número de segundos antes de que se agote el tiempo de espera
- poll_frequency: intervalo de suspensión entre llamadas. Por defecto, es de 0,5 segundos.
- ignore_exceptions: estructura iterable de clases de excepción ignoradas durante las llamadas. De forma predeterminada, contiene únicamente NoSuchElementException.

Ejemplo:

```
deselenium.webdriver.support.esperarimportarWebDriverEsperar

elemento=WebDriverWait(controlador,10).hasta(lambdax:x.encontrar_elemento(Por.IDENTIFICACIÓN,
→"algún ID"))

esta_desaparecido=WebDriverWait(controlador,30,1, (ElementoNotVisibleException)).\

                    hasta_no(lambdax:x.encontrar_elemento(Por.IDENTIFICACIÓN,"algún ID").está_displayed())
```

hasta(*método: Invocable[[D], Literal[Falso] | T],mensaje: cadena = ''*) →t

Llama al método proporcionado con el controlador como argumento hasta que el valor de retorno no se evalúe comoFALSO.

Parámetros

- método -invocable (WebDriver)
- mensaje -mensaje opcional paraExcepción de tiempo de espera

Devoluciones

el resultado de la última llamada a*método*

aumentos

*selenio.excepciones.comunes.TimeoutException*si se agota el tiempo de espera

hasta_no(*método: Invocable[[D], T],mensaje: cadena = ''*) →T | Literal[Verdadero]

Llama al método proporcionado con el controlador como argumento hasta que el valor de retorno se evalúe comoFALSO.

Parámetros

- método -invocable (WebDriver)
- mensaje -mensaje opcional paraExcepción de tiempo de espera

Devoluciones

el resultado de la última llamada a*método*, oVerdaderosi*método*ha planteado una de las excepciones ignoradas

aumentos

*selenio.excepciones.comunes.TimeoutException*si se agota el tiempo de espera

7.31 Compatibilidad con colores

claseselenium.webdriver.support.color.Color(*rojo: cualquiera*,*verde: cualquiera*,*azul: cualquiera*,*alfa: Cualquiera = 1*)

Bases:objeto

Clase de soporte de conversión de color.

Ejemplo:

```
deselenio.webdriver.support.colorimportarColor

imprimir(Color.from_string('#00ff33').rgba) imprimir
(Color.from_string('rgb(1, 255, 3)').maleficio) imprimir
(Color.from_string('azul').rgba)
```

__inicio__(*rojo: cualquiera*,*verde: cualquiera*,*azul: cualquiera*,*alfa: Cualquiera = 1*) →Ninguno

método de clase from_string(*cadena_: cadena*) →*Color*

hexadecimal de propiedad: cadena

propiedad rgb: cadena

propiedad rgba: cadena

7.32 Compatibilidad con WebDriver de activación de eventos

claseselenium.webdriver.support.event_firing_webdriver.EventFiringWebDriver(*conductor:* Controlador web, *escucha_evento:* abstractEventListener)

Bases:objeto

Un contenedor alrededor de una instancia arbitraria de WebDriver que admite eventos de activación.

__inicio__(*conductor:*Controlador web,*escucha_evento:*ResumenEventoEscucha) →Ninguno

Crea una nueva instancia de EventFiringWebDriver.

argumentos

- controlador: una instancia de WebDriver
- event_listener: instancia de una clase que subclasifica AbstractEventListener y la implementa total o parcialmente

Ejemplo:

```
deselenio.webdriverimportarFirefox
deselenium.webdriver.support.eventosimportarEventFiringWebDriver,␣
 →ResumenEventoEscucha

claseMi oyente(AbstractEventListener):
    definiciónantes_navegar_a(ser, URL, controlador):
        imprimir("Antes de navegar a%s"%URL)
    definiciónafter_navigate_to(ser, URL, controlador):
        imprimir("Después de navegar a%s"%URL)

conductor=Firefox()
conductor_ef=EventFiringWebDriver(controlador, MyListener())
ef_driver.conseguir("http://www.google.co.in/")
```

atrás() →Ninguno

cerca() →Ninguno

ejecutar_async_script(*guion*, **argumentos*)

ejecutar_script(*guion*, **argumentos*)

encontrar_elemento(*por = 'identificación'*, *valor=Ninguno*) →*Elemento web*

buscar_elementos(*por = 'identificación'*, *valor=Ninguno*) →Lista[*Elemento web*]

adelante() →Ninguno

conseguir(*URL: cadena*) →Ninguno

abandonar() →Ninguno

propiedad envuelto_driver:*Controlador web*

Devuelve la instancia de WebDriver empaquetada por este EventsFiringWebDriver.

claseselenium.webdriver.support.event_firing_webdriver.EventFiringWebElement(*elemento web:* Elemento web, *controlador_ef:* EventFiring-Controlador web)

Bases:objeto

Un contenedor alrededor de la instancia de WebElement que admite eventos de activación.

__inicio__(*elemento web:*Elemento web, *controlador_ef:*EventFiringWebDriver) →Ninguno

Crea una nueva instancia de EventFiringWebElement.

claro() →Ninguno

hacer clic() →Ninguno

encontrar_elemento(*por = 'identificación'*, *valor=Ninguno*) →*Elemento web*

buscar_elementos(*por = 'identificación'*, *valor=Ninguno*) →Lista[*Elemento web*]

enviar_claves(**valor*) →Ninguno

propiedad elemento_envuelto:*Elemento web*

Devuelve el WebElement envuelto por esta instancia de EventFiringWebElement.

7.33 Soporte de escucha de eventos abstractos

claseselenium.webdriver.support.abstract_event_listener.ResumenEventoEscucha

Bases:objeto

El detector de eventos debe subclasificar e implementar esto total o

parcialmente. after_change_value_of(*elemento*,*conductor*) →Ninguno

después de hacer clic (*elemento*,*conductor*) →Ninguno

después_cerrar(*conductor*) →Ninguno

after_execute_script(*guion*,*conductor*) →Ninguno

después_buscar(*por*,*valor*,*conductor*) →Ninguno

after_navigate_back(*conductor*) →Ninguno

after_navigate_forward(*conductor*) →Ninguno

after_navigate_to(*URL: cadena*,*conductor*) →Ninguno

después_salir(*conductor*) →Ninguno

antes_cambio_valor_de(*elemento*,*conductor*) →Ninguno

antes_hacer clic(*elemento*,*conductor*) →Ninguno

antes_cerrar(*conductor*) →Ninguno

antes_execute_script(*guion*,*conductor*) →Ninguno

antes_buscar(*por*,*valor*,*conductor*) →Ninguno

antes_navigate_back(*conductor*) →Ninguno

antes_navigate_forward(*conductor*) →Ninguno

antes_navigate_to(*URL: cadena*,*conductor*) →Ninguno

antes_salir(*conductor*) →Ninguno

en_excepción(*excepción*,*conductor*) →Ninguno

7.34 Condiciones esperadas Soporte

selenium.webdriver.support.expected_conditions.alerta_está_presente() →Invocable[[*Controlador web*],*Alerta* | Literal[Falso]]

Una expectativa de comprobar si hay una alerta actualmente presente y cambiar a ella.

selenium.webdriver.support.expected_conditions.todo_de(**condiciones_esperadas: Invocable[[D], Literal[Falso] | T]*) →Invocable[[D], Lista[T] | Literal[Falso]]

Una expectativa de que todas las múltiples condiciones esperadas sean verdaderas.

Equivale a un 'Y' lógico. Devuelve: Cuando no se cumple alguna condición esperada: Falso. Cuando se cumplen todas las condiciones esperadas: una lista con el valor de retorno de cada condición esperada.

selenium.webdriver.support.expected_conditions.cualquiera_de(**condiciones_esperadas: Invocable[[D], T]*) → Invocable[[D], Literal[Falso] | T]

Expectativa de que cualquiera de las múltiples condiciones esperadas sea verdadera.

Equivalente a un 'O' lógico. Devuelve resultados de la primera condición coincidente, o False si ninguna lo hace.

selenium.webdriver.support.expected_conditions.elemento_atributo_a_incluir(*locador: Tupla[cadena, cadena], atributo_: cadena*) → Invocable[[*Controlador web* | *Elemento web*], booleano]

Una expectativa para verificar si el atributo dado está incluido en el elemento especificado.

localizador, atributo

selenium.webdriver.support.expected_conditions.element_located_selection_state_to_be(*locador: Tupla[cadena, cadena], está_seleccionado: booleano*) → Invocable[[*Controlador web* | *Nosotros-elemento*], booleano]

Una expectativa de ubicar un elemento y verificar si el estado de selección especificado está en ese estado.

localizador es una tupla de (por, ruta) is_selected es un valor booleano

selenium.webdriver.support.expected_conditions.elemento_ubicado_para_ser_seleccionado(*locador: Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *Elemento web*], booleano]

Se selecciona una expectativa para el elemento a ubicar.

localizador es una tupla de (por, ruta)

selenium.webdriver.support.expected_conditions.elemento_selección_estado_a_ser(*elemento:* Elemento web, *está_seleccionado: booleano*) → Invocable[[Cualquiera], booleano]

Una expectativa para verificar si el elemento dado está seleccionado.

El elemento es el objeto WebElement is_selected es un booleano.

selenium.webdriver.support.expected_conditions.element_to_be_clickable(*marca:* Elemento web / *Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *Elemento web*], Literal[Falso] | *Elemento web*]

Una expectativa para verificar un elemento está visible y habilitada de manera que pueda hacer clic en él. El elemento es un localizador (texto) o un WebElement.

selenium.webdriver.support.expected_conditions.elemento_a_ser_seleccionado(*elemento:* Elemento web) → Invocable[[Cualquiera], bool]

Se selecciona una expectativa para verificar la selección. el elemento es el objeto WebElement

selenium.webdriver.support.expected_conditions.frame_to_be_available_and_switch_to_it(*locador: Tu-ple[cadena, cadena] / cadena*) → Invocable[[*Controlador web*], booleano]

Una expectativa para verificar si el marco dado está disponible para cambiar. Si el marco está disponible, cambia el controlador dado al marco especificado.

selenium.webdriver.support.expected_conditions.invisibilidad_del_elemento(*elemento:* Elemento web / *Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *Elemento web*], *Elemento web* | booleano]

Una expectativa para verificar que un elemento sea invisible o no esté presente en el DOM.

El elemento es un localizador (texto) o un WebElement.

selenium.webdriver.support.expected_conditions.invisibilidad_del_elemento_ubicado(*locador:* Elemento web / *Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *WebEle-mento*], *Elemento web* | booleano]

Una expectativa para verificar que un elemento sea invisible o no esté presente en el DOM.

localizador utilizado para encontrar el elemento

selenium.webdriver.support.expected_conditions.nueva_ventana_está_abierta(*current_handles: Lista[cadena]*) → Invocable[[*Controlador web*], booleano]

La expectativa de que se abra una nueva ventana y aumente el número de identificadores de ventana.

selenium.webdriver.support.expected_conditions.ninguno_de(**condiciones_esperadas: invocable[[D], cualquiera]*) → Invocable[[D], bool]

Expectativa de que ninguna de una o varias condiciones esperadas sea verdadera.

Equivalente a un 'NO-O' lógico. Devuelve un booleano

selenium.webdriver.support.expected_conditions.número_de_ventanas_a_ser(*número_ventanas: int*) → Invocable[[*Controlador web*], booleano]

Expectativa de que el número de ventanas sea un valor determinado.

selenium.webdriver.support.expected_conditions.presencia_de_todos_los_elementos_ubicados(*locador: Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *WebElemento*], Lista[*Elemento web*]]

Expectativa de comprobar que hay al menos un elemento presente en una página web.

El localizador se usa para encontrar el elemento y devuelve la lista de WebElements una vez que se ubican.

selenium.webdriver.support.expected_conditions.presencia_de_elemento_ubicado(*localizador: Tupla[str, cadena]*) → Invocable[[*Controlador web* | *Elemento web*], *Elemento web*]

Una expectativa para verificar que un elemento esté presente en el DOM de una página. Esto no significa necesariamente que el elemento sea visible.

localizador: utilizado para encontrar el elemento, devuelve el WebElement una vez ubicado

selenium.webdriver.support.expected_conditions.estancamiento_de (*elemento:* Elemento web) → Invocable[[Cualquiera], bool]

Espere hasta que un elemento ya no esté adjunto al DOM.

elemento es el elemento a esperar. Devuelve False si el elemento todavía está adjunto al DOM; True en caso contrario.

selenium.webdriver.support.expected_conditions.text_to_be_present_in_element(*locador: Tupla[cadena, cadena], texto_: cadena*) → Invocable[[*Controlador web* | *Elemento web*], booleano]

Una expectativa para verificar si el texto dado está presente en el elemento especificado.

localizador, texto

selenium.webdriver.support.expected_conditions.text_to_be_present_in_element_attribute(*locador: Tuple[cadena, cadena], encontrar_elemento: Invocable[[Controlador web | Nosotros-elemento], booleano]*

Una expectativa para verificar si el texto dado está presente en el atributo del elemento.

localizador, atributo, texto

selenium.webdriver.support.expected_conditions.text_to_be_present_in_element_value(*locador: Tuple[cadena, cadena], texto_: cadena*) → Invocable[[*Controlador web* | *WebElemento*], booleano]

Una expectativa para verificar si el texto dado está presente en el valor del elemento.

localizador, texto

selenium.webdriver.support.expected_conditions.título_contiene(*título: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para verificar que el título contenga una subcadena que distinga entre mayúsculas y minúsculas.

título es el fragmento del título que se espera que devuelva Verdadero cuando el título coincide, Falso en caso contrario

selenium.webdriver.support.expected_conditions.título_es (*título: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para comprobar el título de una página.

title es el título esperado, que debe ser una coincidencia exacta. Devuelve True si el título coincide, false en caso contrario.

selenium.webdriver.support.expected_conditions.cambios_url(*URL: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para verificar la URL actual.

url es la URL esperada, que no debe ser una coincidencia exacta. Devuelve True si la URL es diferente y False en caso contrario.

selenium.webdriver.support.expected_conditions.url_contiene(*URL: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para verificar que la URL actual contenga una subcadena que distinga entre mayúsculas y minúsculas.

URL es el fragmento de URL esperado, devuelve True cuando la URL coincide, False en caso contrario.

selenium.webdriver.support.expected_conditions.coincidencias_url(*patrón: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para verificar la URL actual.

El patrón es el patrón esperado. Esto encuentra la primera aparición del patrón en la URL actual y, como tal, no requiere una coincidencia completa exacta.

selenium.webdriver.support.expected_conditions.url_to_be(*URL: cadena*) →Invocable[[*Controlador web*], booleano]

Una expectativa para verificar la URL actual.

url es la URL esperada, que debe ser una coincidencia exacta y devuelve True si la URL coincide; false en caso contrario.

selenium.webdriver.support.expected_conditions.visibilidad_de(*elemento:* Elemento web) → Invocable[[Cualquiera], Literal[Falso] | *Elemento web*]

Una expectativa para verificar que un elemento, que se sabe que está presente en el DOM de una página, sea visible.

Visibilidad significa que el elemento no solo se muestra sino que también tiene una altura y un ancho mayores que 0. El elemento es WebElement y devuelve el (mismo) WebElement una vez que está visible.

selenium.webdriver.support.expected_conditions.visibilidad_de_todos_los_elementos_ubicados(*locador: Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *WebElemento*], Lista[*Elemento web*] | Literal[Falso]]

Una expectativa para verificar que todos los elementos estén presentes en el DOM de una página y sean visibles. Visibilidad significa que los elementos no solo se muestran sino que también tienen una altura y un ancho mayores que 0.

localizador: utilizado para encontrar los elementos, devuelve la lista de WebElements una vez que están ubicados y son visibles.

selenium.webdriver.support.expected_conditions.visibilidad_de_cualquier_elemento_ubicado(*locador: Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *WebElemento*], Lista[*Elemento web*]]

Una expectativa para comprobar que hay al menos un elemento visible en una página web.

El localizador se usa para encontrar el elemento y devuelve la lista de WebElements una vez que se ubican.

selenium.webdriver.support.expected_conditions.visibilidad_de_elemento_ubicado(*locador: Tupla[cadena, cadena]*) → Invocable[[*Controlador web* | *Elemento web*], Literal[Falso] | *Elemento web*]

Una expectativa para verificar que un elemento esté presente en el DOM de una página y sea visible. Visibilidad significa que el elemento no solo se muestra sino que también tiene una altura y un ancho mayores que 0.

localizador: utilizado para encontrar el elemento, devuelve el WebElement una vez que está ubicado y es visible

CAPÍTULO

OCHO

APÉNDICE: PREGUNTAS FRECUENTES

Otras preguntas frecuentes:https://github.com/SeleniumHQ/selenium/wiki/Preguntas-frecuentes

8.1 ¿Cómo utilizar ChromeDriver?

Descarga lo últimochromedriver desde la página de descarga. Descomprime el archivo:

```
descomprimir chromedriver_linux64.cremallera
```

Deberías ver uncontrolador cromadoejecutable. Ahora puedes crear una instancia de Chrome WebDriver como esta:

```
conductor=controlador web.Chrome (ruta_ejecutable="/ruta/al/chromedriver")
```

El resto del ejemplo debería funcionar como se indica en otra documentación.

8.2 ¿Selenium 2 es compatible con XPath 2.0?

Árbitro:http://seleniumhq.org/docs/03_webdriver.html#how-xpath-works-in-webdriver

Selenium delega las consultas XPath al propio motor XPath del navegador, por lo que Selenium admite XPath compatible con todo lo que admita el navegador. En navegadores que no tienen motores XPath nativos (IE 6,7,8), Selenium solo admite XPath 1.0.

8.3 ¿Cómo desplazarse hasta el final de una página?

Árbitro:http://blog.varunin.com/2011/08/scrolling-on-pages-using-selenium.html

Puedes usar el*ejecutar_script*Método para ejecutar javascript en la página cargada. Por lo tanto, puede llamar a la API de JavaScript para desplazarse hasta la parte inferior o cualquier otra posición de una página.

A continuación se muestra un ejemplo para desplazarse hasta el final de una página:

```
conductor.ejecutar_script("ventana.scrollTo(0, documento.body.scrollHeight);")
```

Elventanaobjeto en DOM tiene undesplazarse haciaMétodo para desplazarse a cualquier posición de una ventana abierta. El desplazamientoAltura es una propiedad común a todos los elementos. El*documento.body.scrollHeight*dará la altura de todo el cuerpo de la página.

8.4 ¿Cómo guardar archivos automáticamente usando el perfil personalizado de Firefox?

Árbitro:http://stackoverflow.com/questions/1176348/access-to-file-download-dialog-in-firefox Árbitro:http://

blog.codecentric.de/en/2010/07/descargas-de-archivos-con-selenium-mission-impossible/

El primer paso es identificar el tipo de archivo que desea guardar automáticamente.

Para identificar el tipo de contenido que desea descargar automáticamente, puede utilizarrizo:

```
rizo-yo URL | grep"Tipo de contenido"
```

Otra forma de encontrar el tipo de contenido es usar elsolicitudesmódulo, puedes usarlo así:

```
importarsolicitudes
tipo de contenido=solicitudes.cabeza('http://www.python.org').encabezados['tipo de contenido']
imprimir(tipo_contenido)
```

Una vez identificado el tipo de contenido, puede usarlo para configurar la preferencia del perfil de Firefox:navegador.helperApps. nuncaAsk.saveToDisk

Aquí hay un ejemplo:

```
importarsistema operativo

deselenioimportarcontrolador web

fp=controlador web.Perfil de Firefox()

fp.establecer_preferencia("navegador.descargar.lista de carpetas",2)
fp.establecer_preferencia("navegador.descargar.manager.showWhenStarting",FALSO) fp.
establecer_preferencia("navegador.descargar.dir", os.obtenercwd())
fp.establecer_preferencia("browser.helperApps.neverAsk.saveToDisk","aplicación/flujo de octetos")

navegador=controlador web.Firefox(firefox_profile=fp)
navegador.conseguir("http://pypi.python.org/pypi/selenio")
navegador.find_element_by_partial_link_text("selenio-2").hacer clic()
```

En el ejemplo anterior,aplicación/flujo de octetosse utiliza como tipo de contenido.

Elnavegador.descargar.dirLa opción especifica el directorio donde desea descargar los archivos.

8.5 ¿Cómo cargar archivos en entradas de archivos?

Seleccione el <tipo de entrada="archivo">elemento y llamar alenviar_claves()método que pasa la ruta del archivo, ya sea la ruta relativa al script de prueba o una ruta absoluta. Tenga en cuenta las diferencias en los nombres de rutas entre los sistemas Windows y Unix.

8.6 ¿Cómo utilizar Firebug con Firefox?

Primero descargue el archivo Firebug XPI, luego llame alagregar_extensiónMétodo disponible para el perfil de Firefox:

```
deselenioimportarcontrolador web

fp=controlador web.Perfil de Firefox()

fp.add_extension(extensión='firebug-1.8.4.xpi')
fp.establecer_preferencia("extensiones.firebug.currentVersion","1.8.4")#Evitar la pantalla de inicio
navegador=controlador web.Firefox(firefox_profile=fp)
```

8.7 ¿Cómo tomar una captura de pantalla de la ventana actual?

Utilice el*guardar_captura de pantalla*método proporcionado por el controlador web:

```
deselenioimportarcontrolador web

conductor=controlador web.Controlador Firefox().conseguir(
'http://www.python.org/') conductor.guardar_captura de
pantalla ('captura de pantalla.png') conductor.abandonar()
```

CAPÍTULO

NUEVE

ÍNDICES Y TABLAS

- geníndice
- modíndice
- buscar

ÍNDICE DEL MÓDULO PYTHON

S

ÍNDICE

Símbolos

F

h

I

oh

PAG

qi

abandonar()

R

S

V

W.

Z