SELENIUM

SELENIUM IDE

Con este ide podemos hacer un recording a una pagina y después revisar la grabación y mirar que fue exactamente lo que hizo por debajo

Vamos al icono de Selenium en la extensión y le damos a

1. New Record
2. Name Project
3. Url: <http://www.atestermate.com/AutomationExample.php>
4. Start recording
5. Cuando terminemos de grabar vamos a la extensión y le damos al cuadro de stop
6. Asignamos un nombre

Ahí vemos las acciones que hicimos durante la grabación

Luego de esto le damos en Run, ahí el programa reproduce la grabación y nos muestra el log de lo que se fue ejecutando de cada uno de nuestros pasos, en este caso todo lo vemos ok porque no hubo ningún error, se pueden cambiar los script y darle mantenimiento.

PROYECTO Configurar selenium

Forma tradicional descargando los jars y forma de Maven usando dependencias de Maven REPOSITORY

PROYECTO TRADICIONAL

1. Nuevo
2. Java Project
3. Project Name: JavaTradicionalSelenium
4. Finalizar

Vamos a selenium.dev

Y descargamos el archivo jar que viene siendo el servidor de selenium, lo guardamos en una ruta para después utilizarlo

1. Damos click en nuevo en JavaTradicionalSelenium, package y lo nombramos
2. En ese package creamos una nueva clase, TestJavaSelenium
3. Boton derecho en el proyecto y damos build-path, configure build-path, librerías, add external jars, buscamos el jar que descargamos, abrir, aplicar y cerrar.
4. Ahora si ponemos el puntero en el WebDriver ya podemos importar o hacer algunas cosas con esa librería ya que la importamos ya.
5. Importamos WebDriver
6. Ya creamos una instancia de webdriver, podemos empezar a codear
7. Si molesta el webdriver vamos al Project, click derecho, buildpath, compiler, java compiler, y cambiamos a compiler level 1.8
8. Debemos descargar el .exe de Chrome, Google download Chrome driver
9. En system.property debemos colocar la ruta del driver de Chrome

Me tocó mover el server de selenium del modulepath al classpath por opciones del proyecto build path y cambiar lo que había en el video por esto:

System.*setProperty*("webdriver.chrome.driver", "C:\\drivers-selenium\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver();

driver.get("http://www.google.com");

Thread.*sleep*(5000);

driver.close();

PROYECTO MAVEN

1. Nuevo
2. Project
3. Maven Project
4. Crear proyecto simple
5. Siguiente
6. Group ID: nombre de la página: com.atestermate
7. ArtifactID: Nombre del proyecto
8. Abrimos el archivo pom.xml, ahí ponemos un tag dependencias
9. Vamos a Google, Maven.repository
10. Buscamos selenium, seleccionamos selenium java
11. Seleccionamos la versión 3.141.59
12. Copiamos la dependencia y la pegamos en el pom.xml, guardamos, con esto ya tenemos en la carpeta Maven dependenceds los jars de selenium de esa librería
13. Click derecho en src/test/java nueva clase, nombre del paquete, de la clase, public
14. Pegamos la misma clase que creamos en el anterior ejercicio y corremos el programa
15. De esta manera no tenemos que descargar nada de ninguna parte, solo usamos las dependencias de Maven y con esto basta

Si queremos ejecutar nuestro proyecto como proyecto de Maven

1. Click en run configuration
2. Click derecho en Maven build, new configuration
3. Ponemos un nombre a la configuración
4. Seleccionamos nuestro proyecto en workspace (mavenprojectselenium)
5. Goals: clean test
6. Aply
7. Run
8. Debe quedar en run configurations, selenium test, en JRE, el jdk 8
9. Descargamos desde Maven repository testng
10. Lo pegamos en el pom.xml
11. Creamos nueva clase en TestSelenium

Para por ejemplo dar click en algún item de la pagina web, vamos al inspector, seleccionams el elemento que queremos que selenium le de click, lo llevamos a eclipse y hacemos lo siguiente:

1. driver.findElement(By.*id*("anchor-modelos-suvs")).click();

de esta manera le estamos diciendo que le de click ahí, porque hasta ahora solo estaba abriendo la pagina nada mas.

Si el elemento no tiene un id entonces le copiamos el xpath

driver.findElement(By.*xpath*("//\*[@id=\'coches-slider\']/div[1]/div/div[13]/div/a[2]")).click();

**REST**

Python tiene 2 librerias importantes para consumir servicios web, Request y urllib2, hoy usaremos request

Usaremos los verbos del protocolo http

Get, post, put y delete

Usamos esta url para hacer las peticiones Httpbin.org

Haremos una petición a Google con método get, get es para obtener un recurso del servidor ósea un archivo json, un archivo xml, etc

1. importamos la librería requests
2. creamos el método necesario para hacer la petición
3. *#importamos requests*
4. *import* requests
6. *if* \_\_name\_\_ == '\_\_main\_\_':
7. *# si if name... geneamos una variable llamada url*
8. url = 'https://www.google.com/?hl=es'
9. *#con la libreria requests importada puedo ejecutar el siguiente metodo*
10. *# que recibe un parametro y sera la url*
11. *# el metodo get nos devuelve un objeto de tipo response, ese objeto lo almacenamos en esta variable*
12. response = requests.get(url)
13. print(response)

Si corremos el programa desde la consola el resultado es

➜ REST python3 main.py

<Response [200]>

Si es 200 es porque la petición se hizo con éxito

Si quiero ver todo el contenido de la petición le doy

*if* response.status\_code == 200:

        print(response.content)

Si necesitamos almacenar el contenido de la pagina web lo haríamos asi:

*if* response.status\_code == 200:

        content = response.content

        file = open('google.html', 'wb')

        file.write(content)

        file.close()

De esta manera obtengo las etiquetas y las almacené en un archivo propio

Ahora vamos a usar una petición usando el método get pero a httpbing

Vamos a usar args y url

Dentro del atributo args vamos a encontrar todos los parámetros que le hayamos enviado al servidor, vamos a enviar desde Python los elementos a url y nombre en args

El método get se encarga de construir la url a partir de los parámetros que le enviemos

url = 'https://httpbin.org/get'

    args = { 'nombre': 'David', 'curso': 'Python', 'nivel': 'intermedio'}

*#con la libreria requests importada puedo ejecutar el siguiente metodo*

*# que recibe un parametro y sera la url*

*# el metodo get nos devuelve un objeto de tipo response, ese objeto lo almacenamos en esta variable*

    response = requests.get(url, params=args)

    print(response.url)

Si queremos obtener un valor del json, por ejemplo origin

*if* response.status\_code == 200:

        response\_json = response.json() *#diccionario*

        origin = response\_json['origin']

        print(origin)

response\_json es un diccionario, luego debemos acceder a el declarando origin y accediendo a response\_json[‘origin’]

Pero también tenemos una librería llamada json y el código quedaría asi:

*#importamos requests*

*import* requests

*import* json

*if* \_\_name\_\_ == '\_\_main\_\_':

url = 'https://httpbin.org/get'

    args = { 'nombre': 'David', 'curso': 'Python', 'nivel': 'intermedio'}

response = requests.get(url, params=args)

    print(response.url)

*if* response.status\_code == 200:

 response\_json = json.loads(response.text)

        origin = response\_json['origin']

        print(origin)

SOAP

SOAP Protocol (Simple Object Acces Protocol)

W3C

WSDL, UDDI

XML

Solo usa el metodo post, cuando hace el request envia el formato en xml, las respuestas son basadas en el formato del WSDL

Estructura del XML

* Envelope (define el iunicio y el fin del mensaje, obligado)
* Header (atributos para el requests,opcional)
* Body (el mensaje)
* Fault (errores que pueden pasar durante el envio)

RET

REST API

* http Protocolo
* Metodos CRUD (post, get, update, delete)
* Codigos de respuesta dependiendo de la transacción (200, 300, 400, 500, )
* XML, Json, Binaries, Text

Estructura

* URI or URL (recurso o endpoint)
* Header (atributos)
* Body (mensaje a mandar atraves del requests)

Porque usar REST API

* Consume menso recursos
* Todos los métodos ayudan con el performance
* Se pueden trabajar varios formatos
* Permite trabajar con otras bases de datos