«Talento Tech»

Automation Testing

Clase 7

Buenos Aires Ciudad

Clase N° 7: Introducción a Selenium WebDriver

Temario

- ¿Qué es Selenium y para qué sirve?
- Instalación del entorno paso a paso
    - Selenium Python (`pip install selenium`)
    - Descarga del WebDriver (ChromeDriver / GeckoDriver)
- Primer script “Hola Selenium” en Sauce Demo (desglose detallado)
- Localizar elementos en https://www.saucedemo.com/inventory.html
- Ejercicios prácticos integrados al proyecto

Objetivos de la clase

En esta clase aprenderás a utilizar Selenium, una poderosa herramienta para automatizar navegadores web. Comenzaremos con la instalación de Selenium y del driver correspondiente al navegador de tu preferencia. A partir de ahí, exploraremos cómo funciona la interacción entre el WebDriver —nuestro “conductor” del navegador— y el propio navegador. Escribiremos un primer script básico que abra una URL, lea el título de la página y cierre la sesión, sentando así las bases para automatizaciones más complejas. También practicaremos el uso de distintos selectores con `find_element`, incluyendo By.ID, By.NAME, By.CSS_SELECTOR y By.XPATH, aplicándolos a elementos reales elegidos por ti. Finalmente, aprenderás a registrar los resultados de tus pruebas con reportes simples utilizando impresiones en consola.

¿Qué es Selenium WebDriver?

Selenium es un framework *open-source* que permite **automatizar navegadores** (como Chrome, Firefox o Edge).

WebDriver es la interfaz que traduce tus comandos en Python (u otros lenguajes) en acciones reales en el navegador: abrir páginas, escribir texto, hacer clic, validar contenido, etc. Es la API que habla con el navegador mediante un driver (ChromeDriver, GeckoDriver, etc.).

💡Imagina que el driver es un **intérprete**: traduce tus comandos Python a instrucciones que el navegador entiende (abrir pestaña, hacer clic, escribir texto).

¿Por qué es clave en QA?

- Replica la interacción del usuario real.
- Detecta regresiones visuales y funcionales.
- Se integra con Pytest, CI/CD, reportes…

Instalación del entorno

Paso 1: Instalar la librería Selenium

Desde consola o terminal:

```
pip install selenium
```

Verificá que se haya instalado correctamente:

```
pip show selenium
```

Asegúrate de tener la versión 4.x (la más moderna al momento de la clase).

```
emilianospinoso@ARJRFFGK3:~/Escritorio/automation_project$ pip show selenium
Name: selenium
Version: 4.32.0
Summary: Official Python bindings for Selenium WebDriver
Home-page: https://www.selenium.dev
Author:
Author-email:
License: Apache 2.0
Location: /home/emilianospinoso/.local/lib/python3.10/site-packages
Requires: certifi, trio, trio-websocket, typing_extensions, urllib3, websocket-client
Required-by:
```

Paso 2: Descargar el WebDriver

Cada navegador necesita su propio "conductor":

Navegador	Driver	Descarga
Chrome	**ChromeDriver**	https://googlechromelabs.github.io/chrome-for-testing/
Firefox	**GeckoDriver**	https://github.com/mozilla/geckodriver/releases
Edge	**msedgedriver**	https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/

⚠️ **La versión del driver debe coincidir con la de tu navegador.** Por ejemplo, si tienes Chrome 123, descarga ChromeDriver 123.

En mi caso, con linux Debian, si ejecuto:
`google-chrome --version`

Veo lo siguiente:
Google Chrome 133.0.6943.53

Es por eso que debo descargar esa version en especifico. Linux nos permite hacerlo de esta manera:

```
wget
https://storage.googleapis.com/chrome-for-testing-public/133.0.6943.53/linux64/chromedriver-linux64.zip
```

Paso 3: Verificar la instalación

Comprobá que todo funcione:
Ejecuta : `chromedriver --version`

Resultado esperado:

```
ChromeDriver                                                133.0.6943.53 (9a80935019b0925b01cc21d254da203bc3986f04-refs/branch-heads/6943@{#1389})
```

Ejecuta:

```
python3 -c "from selenium import webdriver; print('Selenium OK')"
```

Resultado esperado:

```
Selenium OK
```

Si todo está correcto, estás listo para comenzar 🚀

Tu primer script WebDriver con Sauce Demo

Explicación paso a paso:

Vamos a crear un script básico que:

1. Abre el navegador.
2. Ingresa a https://www.saucedemo.com.
3. Completa usuario y contraseña.
4. Valida que entramos correctamente al inventario.
5. Cierra el navegador.

🗨️ *Este flujo representa la base de cualquier prueba de interfaz automatizada:* ***abrir → interactuar → validar → cerrar.***

👉Link al script: https://github.com/emilianospinoso/pre-entrega-automation-testing/blob/main/selenium/primer_script.py

selenium > primer_script.py > ...

```python
from selenium import webdriver          #Importamos la librería que permite controlar el navegador
import time                             #Para hacer pausas visibles (solo demo)

driver = webdriver.Chrome()             #Creamos la instancia del driver → abre una ventana de Chrome vacía

try:
    driver.get('https://www.saucedemo.com')  #Navegamos a la URL de Sauce Demo (pantalla de login)

    print('Título:', driver.title)      #Leemos el título de la pestaña → debería salir "Swag Labs"
    assert driver.title == 'Swag Labs'  #Validamos que el título sea el esperado (asegura que cargó bien)

    time.sleep(2)                       #Pausa de 2 s para que lo veas (luego la quitaremos)
finally:
    driver.quit()                       #Cierre limpio: mata la sesión y la ventana
```

¿Cómo ejecutar el script?

1. **Copia** el código en un archivo `primer_script.py` dentro de tu proyecto.
2. Guardá el archivo como `primer_script.py`.
3. Asegurate de tener `chromedriver` en el PATH o junto al archivo.
4. Abre una terminal en esa carpeta.
5. Asegúrate de que tu *virtualenv* esté activo (o trabaja en el intérprete global).
6. Ejecutá en terminal `python3 primer_script.py` y presiona **Enter**.
7. Verás que se abre Chrome → la página de login → se imprime `Título: Swag Labs` → se cierra.

```
emilianospinoso@ARJRFFGK3:~/Escritorio/automation_project/selenium$ python3 primer_script.py
Título: Swag Labs
```

Si aparece un error tipo `selenium.common.exceptions.WebDriverException`, revisa:

- Que `chromedriver` esté en el PATH y coincida con tu versión de Chrome.
- Que instalaste Selenium 4 (***pip show selenium***).

> 🗨️***Recuerda**: cualquier excepción dentro del bloque `try` saltará al `finally`, cerrando el navegador. ¡No quedan procesos zombis!*

Localizar elementos en la página de inventario *(Sauce Demo)*

Una vez que el login es exitoso, el sitio redirige a: Sauce Demo redirige a https://www.saucedemo.com/inventory.html

Allí se carga una **lista de productos**, y es el escenario ideal para **practicar distintos tipos de selectores** que vimos en clases anteriores (CSS y XPath).

Recuerda que para ingresar a esa página debes pasar antes por el login de https://www.saucedemo.com

¿Por qué es clave esta práctica?

Porque la automatización **no solo implica hacer clics**, sino entender **cómo identificar correctamente los elementos del DOM**, aún si cambian de lugar, color o estructura.

Elementos clave disponibles

Elemento	Selector recomendado	Descripción
Botón menú hamburguesa	`#react-burger-menu-btn (ID)`	Abre el sidebar
Ícono del carrito	`.shopping_cart_link (class)`	Accede al carrito
Contador del carrito	`.shopping_cart_badge`	Muestra cantidad de ítems
Título “Products”	`div.header_secondary_container .title`	Encabezado de la grid
Tarjetas de producto	`div.inventory_item (class)`	Cada producto de la lista
Nombre del 1.º producto	`div.inventory_item_name`	Dentro de la 1.ª tarjeta
Botón Add to cart (1.º producto)	`(//button[contains(@id,'add-to-cart')])[1] (XPath)`	Agrega el ítem

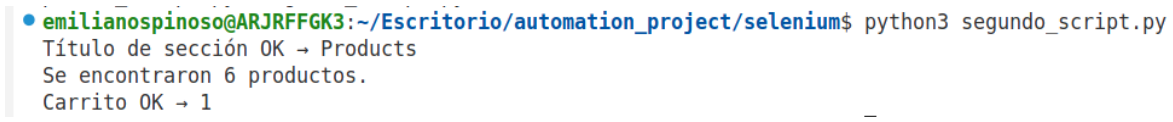
Ejemplo completo: login + validar inventario

Ya lo vimos en el `primer_script.py`, pero estos son ejemplos clave en contexto:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
import time
options = Options()
options.add_argument('--start-maximized')    # Opcional: ventana grande
driver = webdriver.Chrome(options=options)
driver.implicitly_wait(5)  # Espera implícita profesional
try:
   # 1) Login
   driver.get('https://www.saucedemo.com')
   driver.find_element(By.ID, 'user-name').send_keys('standard_user')
   driver.find_element(By.ID, 'password').send_keys('secret_sauce')
   driver.find_element(By.CSS_SELECTOR, 'input[type="submit"]').click()
   # 2) Validar que estamos en inventario
   assert '/inventory.html' in driver.current_url
   # 3) Verificar título de sección
                titulo      =      driver.find_element(By.CSS_SELECTOR,
'div.header_secondary_container .title').text
   assert titulo == 'Products'
   print('Título de sección OK →', titulo)
   # 4) Contar productos visibles
   productos = driver.find_elements(By.CLASS_NAME, 'inventory_item')
   print(f'Se encontraron {len(productos)} productos.')
   # 5) Añadir el primer producto al carrito
   productos[0].find_element(By.TAG_NAME, 'button').click()
   # 6) Confirmar que el badge del carrito muestra 1
                  badge       =       driver.find_element(By.CLASS_NAME,
'shopping_cart_badge').text
   assert badge == '1'
   print('Carrito OK →', badge)
finally:
   driver.quit()
```

¿Qué hace cada bloque?

1. **Configuración**: todo lo referente a ventana y tiempos de espera.
2. **Login**: rellena usuario y contraseña usando IDs y CSS.
3. **Verificación de ruta**: asegura que la URL cambió al inventario.
4. **Lectura de título**: confirma que la página cargó completamente.
5. **Conteo de productos**: recupera todas las tarjetas y muestra cuántas hay.
6. **Añadir al carrito** y **validar el contador**.



Para los que comienzan: copia el script, ejecútalo y **lee la consola**. Verás cada `print()` confirmando los hitos. Si algo falla, Selenium lanzará una excepción señalando la línea exacta.

🗨️**Storytelling**

Automation en *TalentoLab*

Tras varios días de práctica con selectores y estructuras HTML, el equipo de QA se prepara para dar un paso importante: automatizar su primer flujo real de punta a punta. El proyecto de automatización del portal de perfiles está avanzando, y Silvia, como Product Owner, necesita validar que el login y el carrito funcionen correctamente antes de la próxima demo interna.

Ese mismo lunes, durante la daily, llegan las consignas concretas para tu primer script serio con Selenium:

Ejercicios prácticos:

Silvia (PO):

“Queremos una prueba automática que verifique el login y añada un producto al carrito. Todo sobre **Sauce Demo**, para alinear con el proyecto final.”

Matías (Automation Lead):

“Usa Selenium 4, Chrome y los selectores de la clase 6. Imprime ‘Test OK’ si el flujo completo pasa.”

Sube tus scripts en un commit titulado **“Clase 7 – Selenium básico en Sauce Demo”**.

Ejercicios prácticos

Actividad 1 · Script de login

1. Navega a `https://www.saucedemo.com`
2. Ingresa **`standard_user`** / **`secret_sauce`**.
3. Verifica que la URL contenga `/inventory.html`.
4. Imprime **“Test OK”** si pasa.

Reto extra: coloca la **validación del título** justo después de la redirección.

Actividad 2 · Explora el inventario

1. Tras el login, valida que el título (`div.header_secondary_container .title`) sea **“Products”**.
2. Confirma que aparece al menos un `div.inventory_item`.
3. Muestra en consola el nombre y precio del primer producto.

Actividad 3 · Carrito rápido

1. Haz clic en **“Add to cart”** del primer producto.
2. Verifica que el contador del carrito muestre **1**.
3. Navega al carrito y comprueba que el producto añadido está listado.

🛑Estos tres scripts son la base del **Proyecto de Pre-Entrega** (Clase 8).

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad