

Test interface Selenium

- [Test automatisés](#)
- [Programme Python complet](#)

Test automatisés

SIGACS – Documentation du script de démo Selenium

Script de démonstration automatisée de l'interface web SIGACS.

Pilote Firefox en mode graphique, remplit les formulaires caractère par caractère, soumet les données en base, puis marque une pause opérateur après chaque enregistrement.

Prérequis

Outil	Version minimale
Python	3.10+
Firefox	toute version récente
geckodriver	compatible avec Firefox installé
selenium	4.x

Installation rapide :

```
pip install selenium
# geckodriver doit être dans le PATH
```

Lancer le script

```
python demo_sigacs.py
```

Firefox s'ouvre automatiquement, maximisé.

Chaque étape s'affiche dans le terminal.

Après chaque écriture en base, le script se fige jusqu'à ce que l'opérateur appuie sur **Entrée**.

Configuration

Tous les réglages sont regroupés en haut du fichier dans le bloc `CONFIG`.

```
BASE_URL      = "http://192.168.42.131"    # adresse du serveur web

USERNAME      = "demo_user"                # compte existant en base
PASSWORD      = "demo1234"

CHAR_DELAY    = 0.07                       # secondes entre chaque frappe clavier
FIELD_PAUSE   = 1.0                        # pause après avoir rempli un champ
ACTION_PAUSE  = 1.5                        # pause après un clic
SECTION_PAUSE = 2.5                        # pause entre deux étapes majeures
PAGE_WAIT     = 8                          # délai max d'attente d'un élément (WebDriverWait)
```

Modifier un seul nombre change le rythme de toute la démo.

Scénario couvert

Étape	Fonction	Censé échouer
1	<code>test_login</code>	oui
2	<code>test_admin_login</code>	non
1	<code>creation_demo_user</code>	non
1	<code>test_login</code>	non
2	<code>test_new_serre</code>	non
3	<code>test_new_bac</code>	non
4	<code>test_new_culture</code>	non
5	<code>test_logout</code>	non

Pause opérateur

Après chaque soumission validée, le terminal affiche :

```
? New Serre record written - check the DB / UI
? Inspect the result, then press [Enter] to continue...
```

Le navigateur reste ouvert et interactif pendant cette pause.

Appuyer sur **Entrée** reprend le scénario.

Gérer un test censé échouer

Utiliser la fonction utilitaire `attempt()` dans `run_demo()` au lieu d'appeler la fonction directement.

```
def attempt(fn, *args, expect_fail=False, **kwargs):
    try:
        fn(*args, **kwargs)
        if expect_fail:
            print(f"    ? {fn.__name__} était censé échouer mais a réussi")
    except Exception as e:
        if expect_fail:
            print(f"    ? {fn.__name__} a échoué comme prévu : {e}")
        else:
            raise # échec inattendu ? stoppe la démo
```

Exemple d'utilisation dans `run_demo()` :

```
def run_demo() -> None:
    driver = make_driver()
    try:
        attempt(test_login, driver, expect_fail=True) # doit échouer
        attempt(test_admin_login, driver)           # crée le compte
        attempt(test_login, driver)                 # doit réussir
        ...
```

`expect_fail=True` → l'exception est capturée et affichée comme un succès attendu, le scénario continue.

Sans le flag → toute exception remonte et stoppe la démo normalement.

Fonctions utilitaires

Fonction	Rôle
<code>slow_type(element, text)</code>	Tape le texte caractère par caractère avec <code>CHAR_DELAY</code>
<code>pause(duration)</code>	Attend <code>duration</code> secondes (défaut : <code>ACTION_PAUSE</code>)
<code>wait_resume(label)</code>	Affiche le label et attend <code>[Entrée]</code>
<code>wait_for(driver, by, value)</code>	Attend qu'un élément soit présent dans le DOM
<code>wait_clickable(driver, by, value)</code>	Attend qu'un élément soit cliquable
<code>wait_invisible(driver, by, value)</code>	Attend qu'un élément disparaisse
<code>make_driver()</code>	Crée et maximise un driver Firefox headful
<code>attempt(fn, *args, expect_fail)</code>	Exécute un test en capturant les échecs attendus

Programme Python complet

```
"""
SIGACS -- Selenium demo stub
=====
Covers:
  1. Login
  2. Dashboard -- New Serre ? validates (DB write)
  3. Dashboard -- New Bac ? validates (DB write)
  4. Dashboard -- New Culture ? validates (DB write)
  5. Logout

After each DB-write step the script freezes until you
press [Enter] in the terminal -- perfect for a live demo.

Tune every timing constant in the CONFIG block below.
"""

import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.firefox.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# -----
# CONFIG -- change these freely
# -----
BASE_URL      = "http://192.168.42.131"

# Demo credentials (existing account)
ADMIN_USERNAME = "demo_admin"
ADMIN_PASSWORD = "admindemo1234"

USERNAME      = "demo_user"
PASSWORD      = "demo1234"

# Delays (seconds)
CHAR_DELAY    = 0.07 # between each keystroke
FIELD_PAUSE   = 1.0  # after filling a field
ACTION_PAUSE  = 1.5  # after clicking
SECTION_PAUSE = 2.5  # between major steps
PAGE_WAIT     = 8    # max wait for elements

# -----
# HELPERS
# -----

def slow_type(element, text: str) -> None:
    """Type text character by character."""
    element.clear()
    for char in text:
        element.send_keys(char)
        time.sleep(CHAR_DELAY)
```

```

def pause(duration: float = ACTION_PAUSE) -> None:
    time.sleep(duration)

def wait_resume(label: str) -> None:
    """Freeze the demo until the operator presses Enter."""
    print(f"\n    ? {label}")
    print("    ? Inspect the result, then press [Enter] to continue...")
    input()

def wait_for(driver, by, value, timeout=PAGE_WAIT):
    return WebDriverWait(driver, timeout).until(
        EC.presence_of_element_located((by, value))
    )

def wait_clickable(driver, by, value, timeout=PAGE_WAIT):
    return WebDriverWait(driver, timeout).until(
        EC.element_to_be_clickable((by, value))
    )

def wait_invisible(driver, by, value, timeout=PAGE_WAIT):
    """Wait until an element disappears (e.g. modal closes)."""
    return WebDriverWait(driver, timeout).until(
        EC.invisibility_of_element_located((by, value))
    )

def expected_failure(fn):
    def wrapper(*args, **kwargs):
        try:
            fn(*args, **kwargs)
            print(f"    ? {fn.__name__} was expected to fail but passed")
        except Exception as e:
            print(f"    ? {fn.__name__} failed as expected: {e}")
    return wrapper

# -----
# DRIVER SETUP
# -----

def make_driver() -> webdriver.Firefox:
    opts = Options()
    driver = webdriver.Firefox(options=opts)
    driver.maximize_window()
    return driver

# -----
# TEST BLOCKS
# -----

def attempt(fn, *args, expect_fail=False, **kwargs):
    try:
        fn(*args, **kwargs)
        if expect_fail:
            print(f"    ? {fn.__name__} was expected to fail but passed")
    except Exception as e:

```

```

    if expect_fail:
        print(f"      ? {fn.__name__} failed as expected: {e}")
    else:
        raise # échec réel ? plante le demo normalement

def test_admin_login(driver) -> None:
    print("\n--- [1/5] LOGIN ---")
    driver.get(f"{BASE_URL}/login.php")
    pause(SECTION_PAUSE)

    slow_type(wait_for(driver, By.ID, "username"), ADMIN_USERNAME)
    pause(FIELD_PAUSE)

    slow_type(wait_for(driver, By.ID, "password"), ADMIN_PASSWORD)
    pause(FIELD_PAUSE)

    wait_clickable(driver, By.CSS_SELECTOR, "input[type='submit']").click()
    pause(SECTION_PAUSE)

    wait_for(driver, By.CLASS_NAME, "navigation_tree")
    print("      ? Logged in -- dashboard loaded")
    pause(SECTION_PAUSE)

def creation_demo_user(driver) -> None:
    print("\n--- [2/5] creation user ---")

    wait_clickable(driver, By.ID, "add_user").click()
    pause(SECTION_PAUSE)

    driver.get(f"{BASE_URL}/connexions/register.php")
    pause(SECTION_PAUSE)

    slow_type(wait_for(driver, By.ID, "username"), USERNAME)
    pause(FIELD_PAUSE)

    slow_type(wait_for(driver, By.ID, "prenom"), "Utilisateur")
    pause(FIELD_PAUSE)

    slow_type(wait_for(driver, By.ID, "nom"), "De-la-Démo")
    pause(FIELD_PAUSE)

    slow_type(wait_for(driver, By.ID, "password"), PASSWORD)
    pause(FIELD_PAUSE)

    slow_type(wait_for(driver, By.ID, "confirm_password"), PASSWORD)
    pause(FIELD_PAUSE)

    wait_clickable(driver, By.CSS_SELECTOR, "input[type='submit']").click()
    pause(SECTION_PAUSE)

    wait_clickable(driver, By.ID, "log_out").click()
    pause(SECTION_PAUSE)

def test_login(driver) -> None:
    print("\n--- [1/5] LOGIN ---")
    driver.get(f"{BASE_URL}/login.php")
    pause(SECTION_PAUSE)

    slow_type(wait_for(driver, By.ID, "username"), USERNAME)
    pause(FIELD_PAUSE)

```

```

slow_type(wait_for(driver, By.ID, "password"), PASSWORD)
pause(FIELD_PAUSE)

wait_clickable(driver, By.CSS_SELECTOR, "input[type='submit']").click()
pause(SECTION_PAUSE)

wait_for(driver, By.CLASS_NAME, "navigation_tree")
print("    ? Logged in -- dashboard loaded")
pause(SECTION_PAUSE)

def test_new_serre(driver) -> None:
    print("\n--- [2/5] NEW SERRE ---")

    wait_clickable(driver, By.ID, "newSerreBtn").click()
    pause(ACTION_PAUSE)

    slow_type(
        wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='nom']"),
        "Serre Demo"
    )
    pause(FIELD_PAUSE)

    slow_type(
        wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='adresse']"),
        "42 rue de la Demo, Lorient"
    )
    pause(FIELD_PAUSE)

    slow_type(
        wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='surface']"),
        "25.5"
    )
    pause(FIELD_PAUSE)

    slow_type(
        wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='bac']"),
        "3"
    )
    pause(FIELD_PAUSE)

    # --- SUBMIT ---
    wait_clickable(driver, By.ID, "formSubmit").click()
    pause(SECTION_PAUSE)

    print("    ? Serre submitted")
    wait_resume("New Serre record written -- check the DB / UI")

def test_new_bac(driver) -> None:
    print("\n--- [3/5] NEW BAC ---")

    wait_clickable(driver, By.ID, "newBacBtn").click()
    pause(ACTION_PAUSE)

    slow_type(
        wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='nom']"),
        "Bac Demo A"
    )

```

```

pause(FIELD_PAUSE)

slow_type(
    wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='blocX']"),
    "4"
)
pause(FIELD_PAUSE)

slow_type(
    wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='blocY']"),
    "3"
)
pause(FIELD_PAUSE)

slow_type(
    wait_for(driver, By.CSS_SELECTOR, "#modalContainer input[name='numero']"),
    "1"
)
pause(FIELD_PAUSE)

# --- SUBMIT ---
wait_clickable(driver, By.ID, "formSubmit").click()
pause(SECTION_PAUSE)

print("    ? Bac submitted")
wait_resume("New Bac record written -- check the DB / UI")

def test_new_culture(driver) -> None:
    print("\n--- [4/5] NEW CULTURE ---")

    wait_clickable(driver, By.ID, "newCultureBtn").click()
    pause(ACTION_PAUSE)

    # Plant name (first 'nom' input)
    all_nom = driver.find_elements(By.CSS_SELECTOR, "#modalContainer input[name='nom']")
    slow_type(all_nom[0], "Tomate cerise")
    pause(FIELD_PAUSE)

    # Latin name (second 'nom' input)
    if len(all_nom) > 1:
        slow_type(all_nom[1], "Solanum lycopersicum")
        pause(FIELD_PAUSE)

    # Humidity & temperature numeric fields
    fields = {
        "humidite-ambiante (min)": ("humidite-ambiante", 0, "40"),
        "humidite-ambiante (max)": ("humidite-ambiante", 1, "85"),
        "humidite-ambiante (opt)": ("humidite-ambiante", 2, "65"),
        "humidite-sol (min)": ("humidite-sol", 0, "30"),
        "humidite-sol (max)": ("humidite-sol", 1, "80"),
        "humidite-sol (opt)": ("humidite-sol", 2, "60"),
        "temperature (min)": ("temperature-ambiante", 0, "15"),
        "temperature (max)": ("temperature-ambiante", 1, "32"),
        "temperature (opt)": ("temperature-ambiante", 2, "22"),
    }

    for label, (name, idx, val) in fields.items():
        inputs = driver.find_elements(
            By.CSS_SELECTOR, f"#modalContainer input[name='{name}']"

```

```

    )
    if idx < len(inputs):
        slow_type(inputs[idx], val)
        pause(FIELD_PAUSE * 0.5)

# Temps de pousse
temps = driver.find_elements(By.CSS_SELECTOR, "#modalContainer input[name='temps']")
if temps:
    slow_type(temps[0], "90")
    pause(FIELD_PAUSE)

# --- SUBMIT ---
wait_clickable(driver, By.ID, "formSubmit").click()
pause(SECTION_PAUSE)

print("    ? Culture submitted")
wait_resume("New Culture record written -- check the DB / UI")

def test_logout(driver) -> None:
    print("\n--- [5/5] LOGOUT ---")

    pause(ACTION_PAUSE)
    wait_clickable(driver, By.CSS_SELECTOR, "a[href='logout.php']").click()
    pause(SECTION_PAUSE)

    wait_for(driver, By.ID, "username")
    print("    ? Logged out -- back on login page")
    pause(SECTION_PAUSE)

# -----
# MAIN RUNNER
# -----

def run_demo() -> None:
    driver = make_driver()
    try:
        attempt(test_login, driver, expect_fail=True)
        attempt(test_admin_login, driver)
        attempt(creation_demo_user, driver)
        attempt(test_login, driver)
        attempt(test_new_serre, driver)
        attempt(test_new_bac, driver)
        attempt(test_new_culture, driver)
        attempt(test_logout, driver)
        print("\n? Demo complete.")
    except Exception as exc:
        print(f"\n? Demo failed: {exc}")
        raise
    finally:
        pause(SECTION_PAUSE)
        driver.quit()

if __name__ == "__main__":
    run_demo()

```