**Selenium ile Paralel Test Çalıştırma**

Paralel test çalıştırma, test süreçlerini hızlandırmak ve kaynakları verimli kullanmak için birden fazla testi aynı anda çalıştırma özelliğidir. Selenium kullanılarak Python tabanlı testlerin paralel olarak yürütülmesi, doğru yapılandırma ve uygun kütüphanelerle kolaylıkla gerçekleştirilebilir.

**1. Paralel Testlerin Yapılandırılması**

Paralel testler, şu iki popüler Python test framework’ü ile yapılandırılabilir:

* **pytest**: Esnek ve güçlü bir test frameworküdür.
* **unittest**: Python’un yerel test kütüphanesidir.

**pytest ile Paralel Testler**

pytest frameworkü, çeşitli eklentilerle birlikte paralel test desteği sağlar. Bunun için öncelikle pytest-xdist kütüphanesinin kurulu olduğundan emin olun.

pip install pytest-xdist

**unittest ile Paralel Testler**

concurrent.futures modülü yardımıyla unittest frameworkü için paralel test çalıştırma sağlanabilir. unittest içinde manuel bir paralelleştirme uygulanması gereklidir.

**2. pytest-xdist Kullanarak Paralel Testlerin Çalıştırılması**

pytest-xdist, testlerin çoklu çekirdekte ya da farklı makinelerde paralel olarak çalışmasını sağlar.

**Örnek Yapılandırma**

1. Test senaryolarınızı bir Python dosyasına yazın (örneğin: test\_example.py).
2. Aşağıdaki komutla testlerinizi paralel olarak çalıştırın:

pytest -n <çekirdek\_sayısı>

**Kod Örneği:**

import pytest  
from selenium import webdriver  
from selenium.webdriver.chrome.service import Service  
from webdriver\_manager.chrome import ChromeDriverManager  
  
  
  
@pytest.fixture(scope="function")  
def browser():  
 driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))  
 yield driver  
 driver.quit()  
  
def test\_google(browser):  
 browser.get("https://www.samsun.edu.tr")  
 assert "Samsun" in browser.title  
  
def test\_bing(browser):  
 browser.get("https://mf.samsun.edu.tr/")  
 assert "Mühendislik" in browser.title

Komut:

pytest -n 2

*#pytest -n 2 --cache-clear -v paralel\_farkli\_seneryo.py*

Bu komut, testleri iki çekirdekte paralel olarak çalıştırır.

**3. Farklı Test Senaryolarını Aynı Anda Yürütme**

Farklı tarayıcılarda aynı anda test yürütmek için Selenium’un Grid yapısı ya da pytest ile uygun yapılandırmalar kullanılabilir.

**Örnek:** Chrome ve Firefox tarayıcılarında aynı anda test yürütme:

import pytest  
from selenium import webdriver  
from selenium.webdriver.chrome.service import Service as ChromeService  
from selenium.webdriver.firefox.service import Service as FirefoxService  
from webdriver\_manager.chrome import ChromeDriverManager  
from webdriver\_manager.firefox import GeckoDriverManager  
  
@pytest.fixture(scope="function", params=["chrome", "firefox"])  
def browser(request):  
 if request.param == "chrome":  
 driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))  
 elif request.param == "firefox":  
 driver = webdriver.Firefox(service=FirefoxService(GeckoDriverManager().install()))  
 yield driver  
 driver.quit()  
  
def test\_open\_websites(browser):  
 if isinstance(browser, webdriver.Chrome):  
 browser.get("https://yazilimmuhendisligi.samsun.edu.tr/")  
 assert "Yazılım" in browser.title  
 elif isinstance(browser, webdriver.Firefox):  
 browser.get("https://mf.samsun.edu.tr")  
 assert "Mühendislik" in browser.title  
  
  
  
*#pytest -n 2 --cache-clear -v paralel\_farkli\_seneryo.py*

**4. Performans İyileştirmeleri İçin Paralel Test Kullanımı**

* **Kaynak Kullanımı:** Paralel testler, birden fazla çekirdek veya makine kullanılarak sistem kaynaklarını daha etkin kullanır.
* **Süre Azaltma:** Testlerin toplam çalışma sürelerini önemli ölçülerde azaltır.
* **Süreklilik:** CI/CD (Sürekli Entegrasyon ve Sürekli Dağıtım) süreçlerinde testlerin hızlı yürütülmesini sağlar.

@pytest.fixture(scope="function", params=["chrome", "firefox"])

1. **@pytest.fixture**:
   * Bu bir **decorator**dür ve pytest'in bir **fixture** oluşturmasını sağlar. Fixture'lar, test fonksiyonlarına bağımlılık (dependency) sağlayan yöntemlerdir. Testlerde tekrar kullanılabilir veri veya ayarları sağlayabilir.
2. **scope="function"**:
   * Bu, fixture'ın ömrünü belirler.
   * function: Bu durumda, fixture her test fonksiyonu çağrıldığında yeniden çalıştırılır. Yani her test için yeni bir örnek oluşturulur.
   * Alternatif scope değerleri:
     + "module": Bir modül boyunca bir kez çalışır.
     + "class": Sınıf içindeki testler için bir kez çalışır.
     + "session": Test oturumu boyunca bir kez çalışır.
3. **params=["chrome", "firefox"]**:
   * Bu, fixture için farklı parametreler sağlar. **params** listesi içindeki her öğe, ilgili test fonksiyonu için bir test varyasyonu oluşturur.
   * Örnekte, fixture iki tarayıcı için test yapar: "chrome" ve "firefox".
   * Bu, **parametrik testler** yapılmasını sağlar. Yani, aynı test fonksiyonu farklı parametrelerle birden fazla kez çalıştırılır.