Pytest & Playwright

Pytest

Имена файлов тестов должны иметь префикс test_ или постфикс _test. Имена тестов должны иметь префикс test_

В модуле utils функция root, отнимающая 1 от входного параметра. Пример теста:

```
import utils


def test_first():
    num24 = utils.root(25)
    assert num24 == 24
```

Запуск теста:

```
pytest second_test.py
```

Ключ -v отображает расширенную информацию.
Ключ -s разрешает вывод данных из тестируемых функций.

Запуск без указания имени файла исполняет все тесты.

Желательно определение типов в функциях.

Виды проверок

Проверка	Описание
assert mynum == 32	Проверка на значение
assert type(dt) == dict	Проверка типа
assert "timestamp" in dt_list	Проверка присутствия в списке

Для вывода доп. информации:

```
assert mynum == 32, "Число mynum должно быть равно 32"
```

Фикстуры:

Для упаковки повторяющихся действий.

```
import json
import report
import pytest

@pytest.fixtures
def report_json():
    report.generate_report()
    with open("report.json") as file:
        return json.load(file)

def test_report_json(report_json):
    assert type(report_json) == dict
```

Фикстура исполняется каждый раз при вызове. Для однократного исполнения фикстуры нужно добавить scope="session"

```
import json
import report
import pytest

@pytest.fixtures(scope="session")
def report_json():
    report.generate_report()
    with open("report.json") as file:
        return json.load(file)

def test_report_json(report_json):
    assert type(report_json) == dict
```

Виды scope

session	Один раз в пределах сессии
function	Каждый раз
module	Если несколько тестовых файлов, то один раз в пределе запуска.

Pytest-playwright

Данный плагин упрощает работу с pytest. Встроенные фикстуры. Задачу создания и удаления playwright, browser берет на себя.

```
from playwright.sync_api import Page


def test_first(page: Page):
    page.goto("https://playwright.dev/python")
    link = page.get_by_role("link", name="GET STARTED")
    link.click()
    assert page.url == "https://playwright.dev/python/docs/intro"
```

Настройка pytest

Через консоль:

- --headed Отображать окно
- --slowmo=500 Замедление работы
- --browser=firefox Настройка браузера

Через конфигурационный файл pytest.ini

```
[pytest]
addopts = --headed --slowmo=500 --browser=firefox
```

Фикстуры

Добавить фикстуру

```
import pytest
from playwright.sync_api import Page


@pytest.fixture
def load_testpage(page: Page):
    page.goto("https://playwright.dev/python")
    return page


def test_first(load_testpage: Page):
    link = load_testpage.get_by_role("link", name="GET STARTED")
    link.click()
    assert load_testpage.url == "https://playwright.dev/python/docs/intro"
```

Автоматическое исполнение фикстуры перед каждой функцией:

```
import pytest
from playwright.sync_api import Page

@pytest.fixture(autouse=True)
def load_testpage(page: Page):
    page.goto("https://playwright.dev/python")
    return page

def test_first(page: Page):
    link = page.get_by_role("link", name="GET STARTED")
    link.click()
    assert page.url == "https://playwright.dev/python/docs/intro"
```

Пред и пост исполнение кода:

```
import pytest
from playwright.sync_api import Page

@pytest.fixture(autouse=True)
def load_testpage(page: Page):
    page.goto("https://playwright.dev/python")
    yield page
    page.goto("https://yandex.ru")

def test_first(page: Page):
    link = page.get_by_role("link", name="GET STARTED")
    link.click()
    assert page.url == "https://playwright.dev/python/docs/intro"
```

Пример проверки авторизации ВК

```
'''Test for auth module'''
import pytest
from playwright.sync_api import Browser, Page, expect

AUTH_URL = "https://wood.bobrobotirk.ru/auth"

@pytest.fixture
def page_and_auth(browser: Browser):
    context = browser.new_context(
```

```
        storage_state="playwright/.auth/vk.json"
        )
    #page = context.new_page()
    yield context
    context.close()

def get_cookie(all_cookies, cname):
    val = next((cookie['value'] for cookie in all_cookies if cookie.get('name') == cname), None)
    return val

def test_first(page_and_auth: Browser):
    page = page_and_auth.new_page()
    page.goto(AUTH_URL)
    vkframe = page.frame(url=lambda url: "id.vk.com" in url)

    if vkframe:
        authbutton = vkframe.get_by_role("button", name="Продолжить как")
        expect(authbutton, "Кнопка Продолжить как... отсутствует").to_be_visible(timeout=20000)
        authbutton.click()
        back_button = page.get_by_role("button", name="Авторизация успешна")
        expect(back_button, "Сервис не произвел авторизацию").to_be_visible()
        back_button.click()
        all_cookies = page_and_auth.cookies()
        session_id_value = get_cookie(all_cookies, 'session_id')
        assert session_id_value, "Cookie session_id не установлен."
    else:
        assert False, "VK фрейм не найден."
```

Revision #6
Created 18 April 2025 09:00:55 by Admin
Updated 19 April 2025 07:06:02 by Admin