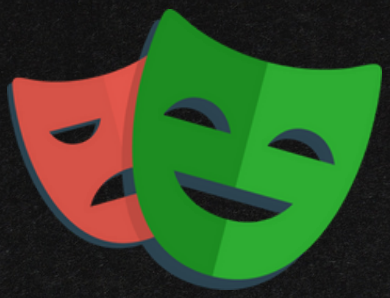


Playwright Test Automation

Tips & Tricks

Improving Playwright tests using the powerful combination of Fixtures & Page Object Model (POM)

Davit Mkhitaryan

Swipe →

Playwright Fixtures

Test fixtures are used to establish the environment for each test, giving the test everything it needs and nothing else.

For example, “page” is a built-in fixture providing your test with an instance of Page object and access to its methods and properties.

```
import { test, expect } from '@playwright/test'

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/')

  await expect(page).toHaveTitle(/Playwright/)
})
```

Playwright Docs - https://playwright.dev/docs/test-fixtures

Davit Mkhitaryan

Swipe →

Page Object Model (POM)

The page object model is a design pattern that helps to separate the page logic from the tests. POM is a class-based object that represents a page. It contains all the methods and properties that are used to interact with a given page.

```ts
import { Page, Locator } from '@playwright/test';

export default class LoginPage {

  readonly page: Page;

  // Locators
  readonly usernameField: Locator;
  readonly passwordField: Locator;
  readonly loginButton: Locator;


  constructor(page: Page) {
    this.page = page;

    // Locators
    this.usernameField = this.page.locator('#user-name');
    this.passwordField = this.page.locator('#password');
    this.loginButton = this.page.locator('#login-button');
  }

  async navigate(): Promise<void> {
    await this.page.goto('/');
  }

  async login(username: string, password: string): Promise<void> {
    await this.usernameField.type(username);
    await this.passwordField.type(password);
    await this.loginButton.click();
  }
}
```

Davit Mkhitaryan

Swipe →

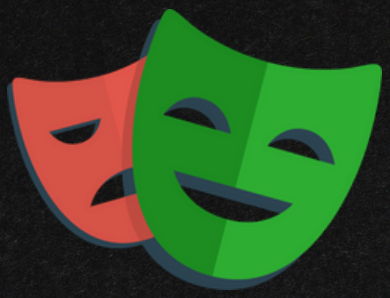
Creating POM fixture

You can create your own fixture in Playwright by extending the base test and use it to create instances of your page classes - as many as you wish. Thus the instance of your page object along with its methods and properties become accessible to your tests.

```ts
import { test as base } from '@playwright/test'
import LoginPage from '../pages/login.page'
import CartPage from '../pages/cart.page'
import InventoryPage from '../pages/inventory.page'

interface Pages {
  loginPage: LoginPage;
  cartPage: CartPage;
  inventoryPage: InventoryPage;
}

export const test =
  base.extend <
  Pages >
  {
    loginPage: async ({ page }, use) => {
      await use(new LoginPage(page))
    },
    cartPage: async ({ page }, use) => {
      await use(new CartPage(page))
    },
    inventoryPage: async ({ page }, use) => {
      await use(new InventoryPage(page))
    },
  }

export const expect = test.expect
```

Davit Mkhitaryan

Swipe →

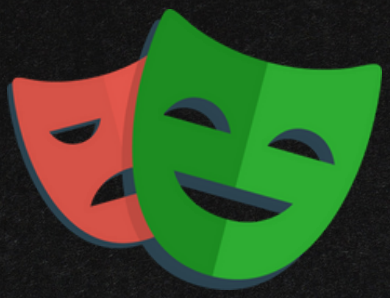


Before using POM Fixture

```
import { expect, test, Page } from '@playwright/test'
import LoginPage from '../pages/login.page'
import InventoryPage from '../pages/inventory.page'
import CartPage from '../pages/cart.page'

test.describe('Demo Tests', () => {
  test('Add single item to shopping cart', async ({ page }: { page: Page }) => {
    const loginPage = new LoginPage(page)
    const inventoryPage = new InventoryPage(page)
    const cartPage = new CartPage(page)

    await loginPage.navigate()
    await loginPage.login('standard_user', 'secret_sauce')
    const titleInventoryPage = await inventoryPage.getInvetoryItemTitleByIndex(
      0
    )
    await inventoryPage.addItemToCartByIndex(0)
    expect(await inventoryPage.getShoppingCartCount()).toEqual('1')
    await inventoryPage.clickShoppingCartIcon()
    expect(await cartPage.getCartItemsCount()).toEqual(1)
    expect(await cartPage.getCartItemTitleByIndex(0)).toEqual(
      titleInventoryPage
    )
  })
})
```

Davit Mkhitaryan

Swipe →

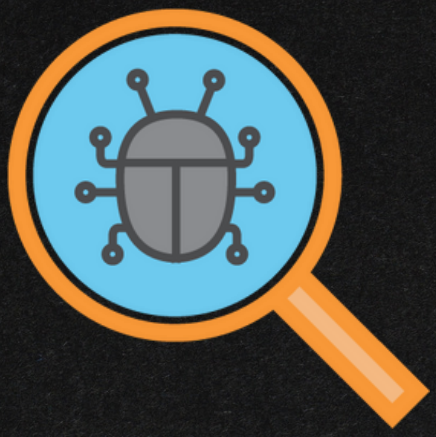


After using POM Fixture

```
import { expect, test } from '../fixtures/pomFixture'

test.describe('Demo Tests', () => {
  test('Add single item to shopping cart', async ({
    loginPage,
    inventoryPage,
    cartPage,
  }) => {
    await loginPage.navigate()
    await loginPage.login('standard_user', 'secret_sauce')
    const titleInventoryPage = await inventoryPage.getInvetoryItemTitleByIndex(
      0
    )
    await inventoryPage.addItemToCartByIndex(0)
    expect(await inventoryPage.getShoppingCartCount()).toEqual('1')
    await inventoryPage.clickShoppingCartIcon()
    expect(await cartPage.getCartItemsCount()).toEqual(1)
    expect(await cartPage.getCartItemTitleByIndex(0)).toEqual(
      titleInventoryPage
    )
  })
})
```

Davit Mkhitaryan

Swipe →



Follow for more useful software testing and test automation content

Davit Mkhitaryan