Fixtures in Playwright Test

**Introduction**

* **Definition**: Test fixtures are essential for establishing the environment needed for each test.
* **Isolation**: Fixtures are isolated between tests, ensuring a clean state.
* **Purpose**: They help group tests by their functional meaning rather than by their setup.

**Built-in Fixtures**

Playwright Test provides several pre-defined fixtures:

| **Fixture** | **Type** | **Description** |
| --- | --- | --- |
| page | Page | An isolated page for the test run. |
| context | BrowserContext | An isolated context for the test run, containing the page fixture. |
| browser | Browser | Shared browsers across tests to optimize resources. |
| browserName | string | Name of the browser currently running the test (chromium, firefox, or webkit). |
| request | APIRequestContext | Isolated API request context for the test run. |

**Without Fixtures**

Using traditional setup methods can be cumbersome. For example:

**TodoPage Class Example**

javascript

Copy code

const { test } = require('@playwright/test');

const { TodoPage } = require('./todo-page');

test.describe('todo tests', () => {

let todoPage;

test.beforeEach(async ({ page }) => {

todoPage = new TodoPage(page);

await todoPage.goto();

await todoPage.addToDo('item1');

await todoPage.addToDo('item2');

});

test.afterEach(async () => {

await todoPage.removeAll();

});

test('should add an item', async () => {

await todoPage.addToDo('my item');

// ...

});

test('should remove an item', async () => {

await todoPage.remove('item1');

// ...

});

});

**With Fixtures**

**Advantages of Using Fixtures**

* **Encapsulation**: Setup and teardown are managed in one place.
* **Reusability**: Fixtures can be defined once and used across multiple test files.
* **On-demand Setup**: Only the necessary fixtures are set up for each test.
* **Composability**: Fixtures can depend on each other for complex setups.
* **Flexibility**: Tests can mix and match fixtures for tailored environments.
* **Simplified Grouping**: Tests can be grouped by functionality instead of setup requirements.

**Example with Custom Fixtures**

javascript

Copy code

import { test as base } from '@playwright/test';

import { TodoPage } from './todo-page';

const test = base.extend<{ todoPage: TodoPage }>( {

todoPage: async ({ page }, use) => {

const todoPage = new TodoPage(page);

await todoPage.goto();

await todoPage.addToDo('item1');

await todoPage.addToDo('item2');

await use(todoPage);

await todoPage.removeAll();

},

});

test('should add an item', async ({ todoPage }) => {

await todoPage.addToDo('my item');

// ...

});

**Creating Custom Fixtures**

* **Definition**: Use test.extend() to create new fixtures.
* **Example**: Create todoPage and settingsPage fixtures.

javascript

Copy code

import { test as base } from '@playwright/test';

import { TodoPage } from './todo-page';

import { SettingsPage } from './settings-page';

type MyFixtures = {

todoPage: TodoPage;

settingsPage: SettingsPage;

};

export const test = base.extend<MyFixtures>({

todoPage: async ({ page }, use) => {

const todoPage = new TodoPage(page);

await todoPage.goto();

await todoPage.addToDo('item1');

await use(todoPage);

await todoPage.removeAll();

},

settingsPage: async ({ page }, use) => {

await use(new SettingsPage(page));

},

});

**Using Fixtures**

* **Integration**: Just mention the fixture in the test function arguments.
* **Example**:

javascript

Copy code

import { test, expect } from './my-test';

test.beforeEach(async ({ settingsPage }) => {

await settingsPage.switchToDarkMode();

});

test('basic test', async ({ todoPage, page }) => {

await todoPage.addToDo('something nice');

await expect(page.getByTestId('todo-title')).toContainText(['something nice']);

});

**Overriding Fixtures**

* **Customization**: You can override existing fixtures.
* **Example**: Automatically navigate to a base URL.

javascript

Copy code

import { test as base } from '@playwright/test';

export const test = base.extend({

page: async ({ baseURL, page }, use) => {

await page.goto(baseURL);

await use(page);

},

});

test.use({ baseURL: 'https://playwright.dev' });

**Worker-scoped Fixtures**

* **Purpose**: Set up fixtures that are shared across tests in the same worker process.
* **Example**: Creating an account fixture.

javascript

Copy code

import { test as base } from '@playwright/test';

export const test = base.extend<{}, { account: Account }>({

account: [async ({ browser }, use, workerInfo) => {

const username = 'user' + workerInfo.workerIndex;

const password = 'verysecure';

// Setup code here...

await use({ username, password });

}, { scope: 'worker' }],

});

**Automatic Fixtures**

* **Definition**: Fixtures set up automatically for each test/worker.
* **Example**: Collecting debug logs on test failure.

javascript

Copy code

import \* as debug from 'debug';

import \* as fs from 'fs';

import { test as base } from '@playwright/test';

export const test = base.extend<{ saveLogs: void }>({

saveLogs: [async ({}, use, testInfo) => {

const logs = [];

debug.log = (...args) => logs.push(args.join(''));

debug.enable('myserver');

await use();

// Log handling...

}, { auto: true }],

});

**Fixture Timeout**

* **Purpose**: Separate timeout for slow fixtures.
* **Example**:

javascript

Copy code

const test = base.extend<{ slowFixture: string }>({

slowFixture: [async ({}, use) => {

await use('hello');

}, { timeout: 60000 }]

});

**Combining Custom Fixtures**

* **Merging**: You can merge fixtures from multiple modules.

javascript

Copy code

import { mergeTests } from '@playwright/test';

import { test as dbTest } from 'database-test-utils';

import { test as a11yTest } from 'a11y-test-utils';

export const test = mergeTests(dbTest, a11yTest);

**Box Fixtures**

* **Purpose**: Reduce noise in reports by boxing fixtures.

javascript

Copy code

import { test as base } from '@playwright/test';

export const test = base.extend({

helperFixture: [async ({}, use) => {

// ...

}, { box: true }],

});

**Custom Fixture Title**

* **Purpose**: Assign a custom title to fixtures for reports.

javascript

Copy code

import { test as base } from '@playwright/test';

export const test = base.extend({

innerFixture: [async ({}, use) => {

// ...

}, { title: 'my fixture' }],

});

**Global BeforeEach/AfterEach Hooks**

* **Implementation**: Use automatic fixtures for global hooks.

javascript

Copy code

import { test as base } from '@playwright/test';

export const test = base.extend<{ forEachTest: void }>({

forEachTest: [async ({ page }, use) => {

await page.goto('http://localhost:8000');

await use();

console.log('Last URL:', page.url());

}, { auto: true }],

});