Clean and Scalable Frontend Tests

with Playwright Fixtures

ARCHITECTING
YOUR
BUSINESS

PENTACOR.DE

- IT ARCHITECTURE
- CLOUD–NATIVE SOFTWARE
- APIS & DATA PRODUCTS

Let‘s connect!

Michael Schlimbach

Software Engineer

michael.schlimbach@pentacor.de

+49 176 561 0 516 7

Goal of this workshop

Learn how to build clean, scalable, and maintainable frontend test suites

COMMON CHALLENGES IN FRONTEND E2E TESTING

Too much boilerplate

Duplicated logic makes tests harder to maintain and slower to write

Fragile tests

Tightly coupled tests break easily when the UI changes

Low reusability

Without modular design, tests are harder to extend and adapt.

OUR LEARNING PATH:

Setup

If not already done, prepare your machine to be ready for the workshop

Syncing up: Write a Playwright test

Refresh your knowledge about the Playwright Syntax and structure

Syncing up: Page Object Models

Refresh your knowledge about the Page Object Model and how it is implemented with Playwright

Exercise: Create first fixtures

Learn how to use fixtures for page objects and remove unnecessary lines of code in your test to make it more readable

Exercise 4: Apply your knowledge

With your new knowledge, create a new test.

Setup Instructions, Slides and Exercises:

https://github.com/SixDollarMan/playwright-fixture-workshop/

```
git checkout workshop/dev-day-2025
```

Turn off CoPilot please (for a better learning experience)

Case Study

Writing tests for XYZ Bank

Scenario

- We are developing tests for a new banking software. The software is still in development, we didn't start implementing tests in the beginning, so the goal is to reduce technical debt.

Disclaimer:

- Example app not perfect for state-of-the-art selectors
- We will need an explicit wait on one point

Exercise I

Syncing up: Playwright Syntax

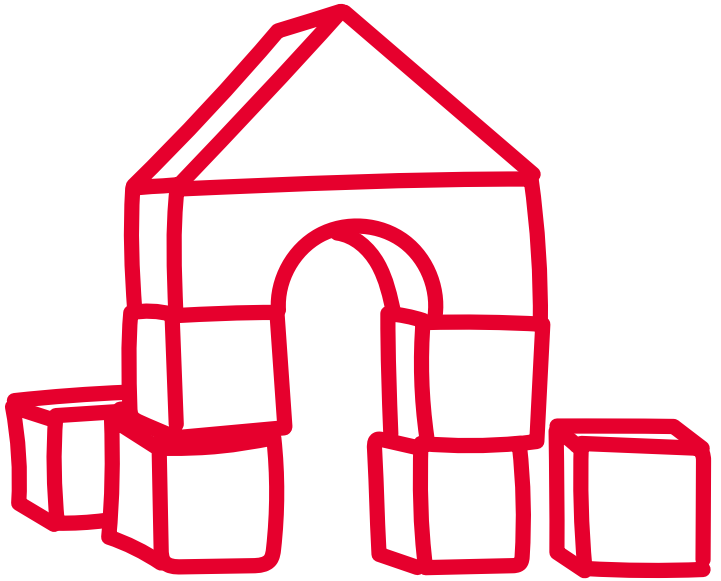
- Open `exercises/exercise1.md` and follow the instructions

The Page Object Model

Encapsulating UI interactions into reusable objects

Benefits

- Improves readability, maintainability and reusability
- Reduces duplication in test logic



Limitations:

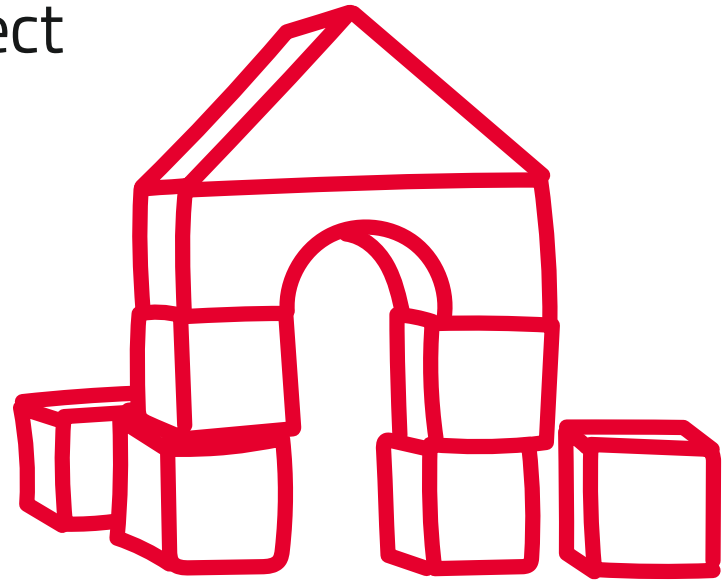
- Can still involve redundant setup logic
- There are possibilities for even better readability like BDD

The Page Object Model

Make tests more human readable

How to achieve this

- Implement page logic as functions into the page object
- Explicitly name them after their task



Examples:

- `await loginPage.loginAs('superadmin')`
- `await checkoutPage.proceedToPayment()`

Exercise II

Syncing up: Page Object Model

- Open `exercises/exercise2.md` and follow the instructions

Coffee break

Playwright Fixtures

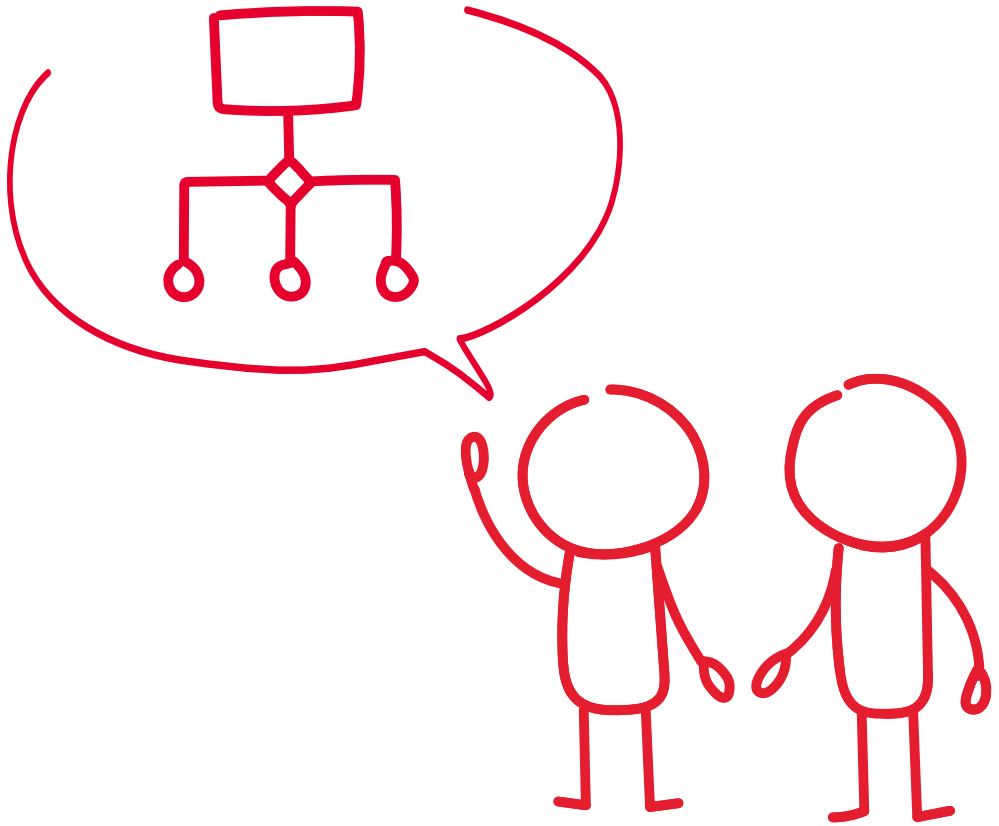
Structured way to manage test setup and teardown logic

Benefits

- Eliminates redundant setup logic
- Enhances test isolation
- Improves resource management

Limitations:

- Risk of unnecessary complexity



Playwright Fixtures

Combining Fixtures with Page Object Model

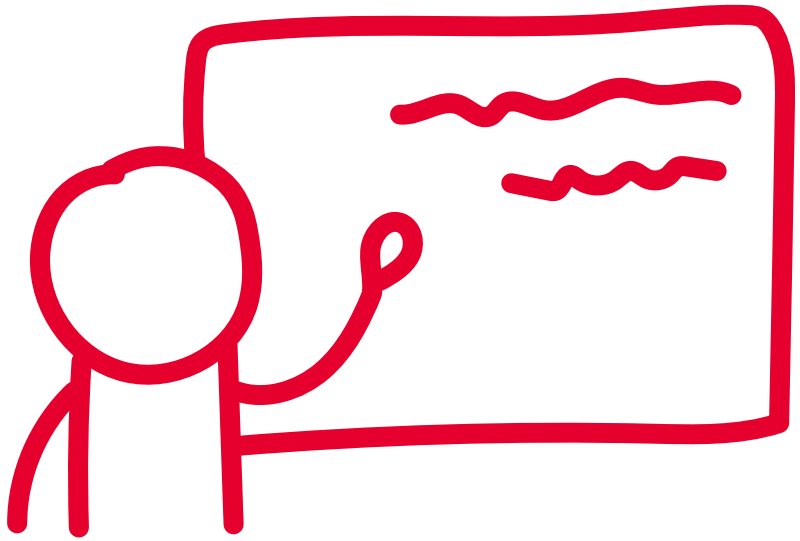
How it works

- Use fixtures to create modular, dependency-injected Page Objects
- Simplify test setup

Key benefits:

- Reduces boilerplate code
- Enhances maintainability

Fixture Demo



Exercise III

Create page object fixtures

- Open `exercises/exercise3.md` and follow the instructions

Fixture Scopes

Providing the right fixture at the right time

Two different scopes

- test-scoped fixtures: ensuring isolation between tests
- worker-scoped fixtures: sharing resources efficiently

Usage:

- Which use case do you see for the different scopes in our case study?

Fixture Learnings

Dos and Don’ts

Dos

- Use test scope for isolation
- Use worker scope for efficiency
- Keep fixtures leightweight
- Make use of the auto teardown
- Documentation!

Fixture Learnings

Dos and Don’ts

Don’ts

- Avoid global state in test-scoped fixtures
- Don’t misuse worker scope for UI states
- Don’t overcomplicate
- Don’t ignore test parallelization

Exercise IV

Create page object fixtures

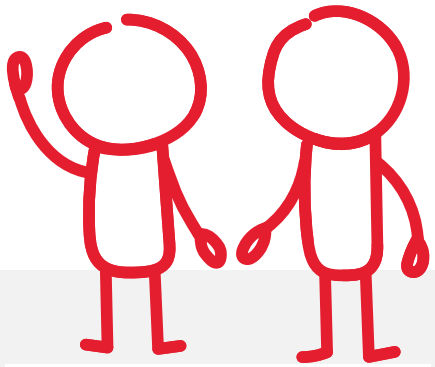
- Open `exercises/exercise4.md` and follow the instructions

Final question

What did you take out of this workshop?

Thank you!
Questions?

Chemnitz
European Capital
of Culture
2025



I appreciate your feedback!