Python+Playwright+Pytest+BDD，用FSM打造高效测试框架（2）

点击蓝字，立即关注

上篇文章那个为大家介绍了**有限状态自动机（FSM）理论基础、FSM在自动化测试中的应用场景及项目结构设计**。

文章后续将继续为大家介绍**复杂业务流程FSM建模与实现及FSM在自动化测试中的进阶应用**等内容。

复杂业务流程FSM建模与实现

5.1 业务流程分析

以“用户注册→邮箱验证→登录→退出”为例，设计如下FSM：

- 状态
  - 未注册
  - 已注册未验证
  - 已验证未登录
  - 已登录

- 事件
  - 注册
  - 邮箱验证
  - 登录
  - 退出

- 状态转移图

5.2 FSM代码实现

```python
# fsm/user_fsm.py
class UserFSM:
    def __init__(self):
        self.state = "未注册"
```

```
    def register(self):
        if self.state == "未注册":
            self.state = "已注册未验证"

    def verify_email(self):
        if self.state == "已注册未验证":
            self.state = "已验证未登录"

    def login(self):
        if self.state == "已验证未登录":
            self.state = "已登录"

    def logout(self):
        if self.state == "已登录":
            self.state = "已验证未登录"
```

5.3 页面对象实现

基础页面类

```
# pages/base_page.py
from playwright.sync_api import Page

class BasePage:
    def __init__(self, page: Page):
        self.page = page
```

注册页面

```
# pages/register_page.py
from .base_page import BasePage

class RegisterPage(BasePage):
    def goto(self):
        self.page.goto("https://example.com/register")

    def register(self, username, email, password):
        self.page.fill("#username", username)
        self.page.fill("#email", email)
        self.page.fill("#password", password)
        self.page.click("#register-btn")
```

登录页面

```
# pages/login_page.py
from .base_page import BasePage

class LoginPage(BasePage):
    def goto(self):
```

```
        self.page.goto("https://example.com/login")

    def login(self, username, password):
        self.page.fill("#username", username)
        self.page.fill("#password", password)
        self.page.click("#login-btn")
```

用户中心页面

```
# pages/dashboard_page.py
from .base_page import BasePage

class DashboardPage(BasePage):
    def logout(self):
        self.page.click("#logout-btn")
```

5.4 Pytest Fixture配置

```
# conftest.py
import pytest
from playwright.sync_api import sync_playwright

@pytest.fixture(scope="function")
def browser_page():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        page = browser.new_page()
        yield page
        browser.close()
```

5.5 测试用例实现

```
# tests/test_user_flow_fsm.py
import pytest
from fsm.user_fsm import UserFSM
from pages.register_page import RegisterPage
from pages.login_page import LoginPage
from pages.dashboard_page import DashboardPage

def test_user_registration_flow(browser_page):
    fsm = UserFSM()
    register_page = RegisterPage(browser_page)
    login_page = LoginPage(browser_page)
    dashboard_page = DashboardPage(browser_page)

    # 1. 未注册 -> 注册
    assert fsm.state == "未注册"
```

```
    register_page.goto()
    register_page.register("testuser", "test@example.com", "password123")
    fsm.register()
    assert fsm.state == "已注册未验证"

    # 2. 邮箱验证（假设自动完成）
    # 这里可以模拟邮箱验证的操作
    fsm.verify_email()
    assert fsm.state == "已验证未登录"

    # 3. 登录
    login_page.goto()
    login_page.login("testuser", "password123")
    fsm.login()
    assert fsm.state == "已登录"

    # 4. 退出
    dashboard_page.logout()
    fsm.logout()
    assert fsm.state == "已验证未登录"
```

5.6 断言与流程控制

在每一步操作后，均通过FSM的状态断言，确保流程的正确性。这种方式不仅验证了页面操作的有效性，也验证了业务流程的完整性。

FSM在自动化测试中的进阶应用

6.1 多分支流程建模

实际业务往往存在多分支、多异常路径。例如，注册时可能遇到用户名已存在、邮箱格式错误等情况。此时，只需在FSM中增加相应状态和转移即可：

```
class UserFSM:
    def __init__(self):
        self.state = "未注册"

    def register(self, username_exists=False, email_invalid=False):
        if self.state == "未注册":
            if username_exists:
                self.state = "注册失败_用户名已存在"
            elif email_invalid:
                self.state = "注册失败_邮箱格式错误"
            else:
                self.state = "已注册未验证"
```

6.2 测试用例自动生成

结合FSM模型，可以自动生成覆盖所有状态转移的测试用例，提升测试覆盖率。例如，利用图遍历算法（如DFS、BFS）遍历所有可能路径，自动生成测试脚本。

示例：自动生成测试路径

```
def dfs(fsm, path, all_paths):
    if fsm.state == "终止状态":
        all_paths.append(list(path))
        return
    for event in fsm.get_possible_events():
        fsm_copy = copy.deepcopy(fsm)
        fsm_copy.trigger(event)
        path.append(event)
        dfs(fsm_copy, path, all_paths)
        path.pop()
```

6.3 与BDD结合

FSM与行为驱动开发（BDD）天然契合。可将每个状态转移映射为Gherkin语法的Step，实现业务流程与测试代码的无缝对接。

示例：Gherkin语法

```
Feature: 用户注册与登录流程
  Scenario: 正常注册、验证、登录、退出
    Given 用户未注册
    When 用户注册
    And 用户邮箱验证
    And 用户登录
    Then 用户已登录
    When 用户退出
    Then 用户已验证未登录
```

6.4 状态覆盖与路径覆盖

- **状态覆盖**：确保每个状态都被测试用例覆盖到。
- **转移覆盖**：确保每个状态转移都被测试用例覆盖到。

- **路径覆盖**：确保所有可能的状态路径都被测试用例覆盖到（通常只对关键路径做覆盖，避免组合爆炸）。

6.5 FSM与数据驱动测试结合

FSM可与数据驱动测试（DDT）结合，实现同一流程下多组数据的自动化测试。例如，注册流程下测试不同的用户名、邮箱、密码组合。

经验分享

7.1 何时引入FSM

- 业务流程复杂、状态众多时，建议引入FSM。
- 流程经常变更、需频繁扩展时，FSM能极大提升可维护性。
- 需自动生成测试用例、提升覆盖率时，FSM是理想选择。

7.2 FSM与POM结合

- FSM负责流程建模，POM负责页面操作。
- 测试用例只需关注流程和断言，页面细节交由POM处理。

7.3 状态与断言分离

- SM只负责状态流转，不直接操作页面。
- 页面操作与断言通过POM实现，保持职责单一。

7.4 代码复用与扩展

- 页面对象可复用于不同测试用例。
- FSM模型可扩展至更多业务场景，如支付、订单、权限等。

7.5 团队协作与文档化

- FSM模型可作为业务流程文档，便于团队沟通和协作。
- 建议用状态转移图、表格等方式可视化FSM，提升可读性。

.....

本文节选自第八十六期《51测试天地》

原创文章

《【测试干货】有限状态自动机（FSM）在自动化测试框架中的实战》

文章后续将继续为大家介绍

“完整项目实战案例及总结”等内容

想继续阅读全文或查看更多《51测试天地》的原创文章

请点击下方 阅读原文或扫描二维码 查看

每日有奖互动

你在测试复杂业务流程时，通常采用什么方法进行建模？

是否尝试过FSM？