Testium

#e2e_sushi

自己紹介

azu
@azu_re

Web scratch, JSer.info

概要

- Testiumって何？
- 同期的なWebDriver API
- PageObjectパターン

Testium

- グルーポンが作ってるE2Eテストフレームワーク
- 同期的なWebDriver APIが特徴

同期的なWebDriver

- `click()` などのブラウザ操作が同期的な処理になってる
- groupon-testium/webdriver-http-sync
- testiumのbrorser APIはこれをラップしたもの
- Nodeには同期HTTP APIがないのでrequest-syncを使ってる

なぜ同期的(なぜ非同期)?

- 非同期だとコールバック地獄が多発
- それを解消するためにPromiseやメソッドチェーンを多様する
  - Jasmineの`expect(promise)`のような特殊なassertが必要
  - テストフレームワークが密接になってしまう
- そもそもUIの操作が非同期であって嬉しい感じではない...

coding-kata/todo-app-jquery-to-backbone

E2Eテストを使ったリファクタリング
のサンプル

素で書いたE2Eテスト

```
var injectBrowser = require('testium/mocha');
var assert = require("power-assert");
var browser;
function addTodo(text) {
    browser.setValue('.todoText', text);
    browser.click('.todoBtn');
}
describe("app-test", function () {
    var text = 'todo text';
    before(injectBrowser());
    beforeEach(function () {
        browser = this.browser;
        this.browser.navigateTo("/");
    });
    context("when テキストボックスに文字を入れて送信した時", function () {

        beforeEach(function () {
            addTodo(text)
        });
        it("should li要素が作成されている", function () {

            var list = browser.getElements('.todoList li');
            assert(list.length > 0);
        });

        it("should リストアイテムのテキストは送信したものと一致している", function () {

            browser.assert.elementHasText('.todoList li', text)
        });
    });
});
```

素で書いたE2Eテスト

- セレクタが直接出てくる
- アプリをリファクタリングして構造が変わると一瞬で壊れる
- 画面や操作を抽象化するリファクタリングが必要
- E2Eテストのリファクタリング => PageObjectパターン?

PageObjectパターン

- The public methods represent the services that the page offers（publicメソッドは、ページが提供するサービスを表す）
- Try not to expose the the internals of the page （ページの内部を公開しないこと）

.... PageObjectデザインパターンを利用して画面変更に強いUIテストを作成する

PageObjectで書きなおしたもの

AppのPageObject

```
function AppPage(browser) {
    this.browser = browser;
    this.browser.navigateTo("/");// Constructorで移動
}
AppPage.prototype.addTodo = function (text) {
    this.browser.setValue('.todoText', text);
    this.browser.click('.todoBtn');// Todoを追加
};
AppPage.prototype.getTodoItems = function () {
    return this.browser.getElements('.todoList li');
};
AppPage.prototype.toggleTodo = function (todo) {
    var input = todo.getElement('input[type="checkbox"]');
    input.click();
};
AppPage.prototype.removeTodo = function (todo) {
    var input = todo.getElement('.removeBtn');
    input.click();
};
module.exports = AppPage;
```

PageObjectを使ったテスト

```
var AppPage = require("./page-objects/app-page");
describe("app-test", function () {
    var inputText = 'todo text';
    var page;
    before(injectBrowser());
    beforeEach(function () {
        // ページの遷移をnew Pageで表せる
        page = new AppPage(this.browser);
    });
    context("when テキストボックスに文字を入れて送信した時", function () {
        beforeEach(function () {
            page.addTodo(inputText)
        });
        it("should li要素が作成されている", function () {
            var list = page.getTodoItems();
            assert(list.length === 1);
        });

        it("should リストアイテムのテキストは送信したものと一致している", function () {
            var todo = page.getTodoItems()[0];
            var text = todo.get("text");
            assert.equal(text, inputText);
        });
    });
})
```

PageObject?

- PageのObject と ComponentのObjectがある
  - Componentで分けても良さそう
- `new Page(driver)` が必ずしも画面遷移ではない