Bài 11
Testing

Mục tiêu

- Trình bày được kiểm thử tích hợp
- Trình bày được kỹ thuật Unit Test
- Triển khai được kiểm thử tích hợp cho ứng dụng ReactJS
- Sử dụng được Unit Test khi kiểm thử ứng dụng ReactJS với Jest

Thảo luận

Tìm hiểu về kiểm thử tích hợp (Integration Testing)

Kiểm thử tích hợp là gì?

- Kiểm thử tích hợp (Integration testing) hay còn gọi là tích hợp và kiểm thử (integration and testing, viết tắt: I&T) là một giai đoạn trong kiểm thử phần mềm.
- Kiểm thử tích hợp xảy ra sau kiểm thử đơn vị (Unit Test) và trước kiểm thử xác nhận.
- Kiểm thử tích hợp nhận các module đầu vào đã được kiểm thử đơn vị, nhóm chúng vào các tập hợp lớn hơn, áp dụng các ca kiểm thử đã được định nghĩa trong kế hoạch kiểm thử tích hợp vào tập hợp đó, và cung cấp đầu ra cho hệ thống tích hợp.

Tại sao lại phải thực hiện kiểm thử tích hợp?

- Một Module nói chung được thiết kế bởi một lập trình viên có hiểu biết và logic lập trình có thể khác với các lập trình viên khác. Kiểm thử tích hợp là cần thiết để đảm bảo tính hợp nhất của phần mềm.
- Tại thời điểm phát triển module, sẽ có nhiều lúc khách hàng đưa ra những thay đổi về yêu cầu, những thay đổi này có thể không được kiểm tra ở giai đoạn unit test trước đó.
- Giao diện và cơ sở dữ liệu của các module có thể chưa hoàn chỉnh khi được ghép lại.
- Khi tích hợp hệ thống các module có thể không tương thích với cấu hình chung của hệ thống.
- Xử lý những trường hợp ngoại lệ một cách không phù hợp có thể gây ra các vấn đề khác

Thảo luận

Tìm hiểu Unit Test và sử dụng Jestjs trong Unit Test

Unit Test là gì?

- Unit Test (hay kiểm thử đơn vị) là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử.
- Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng.
- Mục tiêu của Kiểm thử đơn vị là cô lập một phần code và xác minh tính chính xác của đơn vị đó.
- Các đoạn mã Unit Test hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó Unit Test còn được gọi là kỹ thuật kiểm nghiệm tự động

Unit Test là gì?

- Unit Test có các đặc điểm sau:
  - Đóng vai trò như những người sử dụng đầu tiên của hệ thống.
  - Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

- Unit Test có 3 trạng thái cơ bản:
  - Fail (trạng thái lỗi)
  - Ignore (tạm ngừng thực hiện)
  - Pass (trạng thái làm việc)

Tìm hiểu về Jest

- Jest là một trong những trình chạy thử nghiệm phổ biến nhất hiện nay và đang là lựa chọn mặc định cho các dự án React, Jest có các đặc tính sau:
  - **Đơn giản, dễ hiểu**: bạn ko cần phải đi mò giữa nhiều thư viện khác nhau, chỉ lên trang chủ Jest là đủ
  - **Không cần cấu hình gì cả**: vâng, hoàn toàn không. Chỉ cần kéo thư viện về là bạn có thể bắt đầu viết test và test code được rồi
  - **All in one**: một mình Jest là đã cân đầy đủ test runner, assert và mock. Ngoài ra còn có thêm cả Coverage reports....
  - **Nhanh**: phải nói là rất nhanh, ngoài ra terminal của test rất đẹp và thân thiện

Demo

Thực hành Unit test với Jest

Thảo luận

Test Component trong React với Jest

Cài đặt Jest

- Cài đặt với Create React App

```
yarn add --dev react-test-renderer
```

- Cài đặt không sử dụng Create React App

```
yarn add --dev jest babel-jest @babel/preset-env
@babel/preset-react react-test-renderer
```

- Cài đặt Enzyme

```
npm install –save-dev enzyme enzyme-adapter-react-
16
```

Viết test case đầu tiên

- Tạo một nút đơn giản có tên Click Me bằng đoạn mã sau

```
import React, { useState } from 'react';
import './App.scss';

const App = () =>{
    return (
      <div>
        <button id="ClickMe" className="click-me">Click Me</button>
      </div>
    )
}
export default App;
```

Viết test case đầu tiên

- Thêm đoạn mã sau vào tệp App.test.tsx, đây là tệp viết các trường hợp thử nghiệm

```
import React from 'react'
import Enzyme, { shallow } from 'enzyme'
import Adapter from 'enzyme-adapter-react-16'
import App from './App’

Enzyme.configure({ adapter: new Adapter() })

describe('Test Case For App', () => {
  it('should render button', () => {
    const wrapper = shallow(<App />)
    const buttonElement  = wrapper.find('#ClickMe');
    expect(buttonElement).toHaveLength(1);
    expect(buttonElement.text()).toEqual('Click Me');
  })
})
```

Viết test case đầu tiên

- Sử dụng lệnh sau để chạy các trường hợp kiểm tra

  `npm test`

```
D:\React Projects\sample-test-case-application>npm test

> sample-test-case-application@0.1.0 test D:\React Projects\sample-test-case-application
> react-scripts test
 PASS  src/App.test.tsx
  Test Case For App
    √ should render button (14ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.108s, estimated 6s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```

Demo

Thực hành viết test case với Jest

Tổng kết

Qua bài này chúng ta đã tìm hiểu:

- Kiểm thử tích hợp (Integration Testing)
- Unit Test và sử dụng Jestjs trong Unit Test
- Test Component trong React với Jest

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp theo: Deploy dự án ReactJs