



Unit Test Coverage

Khám phá tiêu chuẩn và best practices cho Unit Test Coverage.

Thành viên:

- 22120026 - Phan Minh Gia Bảo
- 22120035 - Dương Thiện Chí
- 22120047 - Trần Xuân Đăng

01

Loại Code Coverage

Code coverage là một chỉ số đo lường mức độ mã nguồn của một chương trình được kiểm tra, có các loại coverage như:

- Line Coverage
- Branch Coverage
- Function Coverage
- Path Coverage

Line Coverage

Line coverage là một loại chỉ số trong code coverage, đo lường tỷ lệ các dòng mã nguồn được thực thi bởi các bài kiểm tra tự động. Cụ thể, nó cho biết bao nhiêu phần trăm các **dòng** mã trong chương trình đã được kiểm tra.

Line coverage **cao** cho thấy nhiều dòng mã **đã được kiểm tra**, nhưng không đảm bảo rằng tất cả các trường hợp sử dụng và điều kiện đã được kiểm tra.

Line coverage được tính bằng cách **chia số dòng mã đã được thực thi cho tổng số dòng mã** trong chương trình, sau đó nhân với 100 để ra phần trăm.

$$\text{Code coverage (test coverage)} = \frac{\text{Lines of code executed}}{\text{Total number of lines}}$$

Các công cụ như JaCoCo (Java), Istanbul (JavaScript), và Coverage.py (Python)

LCOV - code coverage report

Current view: [top level](#) - [src/routers/routers.dart](#) - routers.dart (source / functions)
Test: lcov.info
Date: 2022-06-19 21:52:04

Line data	Source code
1	: import 'package:go_router/go_router.dart';
2	: import 'package:moxify_app/src/features/features.dart';
3	:
4	4 : final router = GoRouter(
5	: debugLogDiagnostics: true,
6	6 : routes: [
7	7 : GoRoute(
8	: name: 'home',
9	: path: '/',
10	10 : builder: (context, state) => const HomePage(),
11	11 :),
12	12 : GoRoute(
13	: name: 'login',
14	: path: '/login',
15	15 : builder: (context, state) => const LoginPage(),
16	16 :),
17	17 :],
18	:);

Generated by: LCOV version 1.15.alpha0w

Branch Coverage

Branch coverage là một loại chỉ số trong code coverage, đo lường tỷ lệ các nhánh trong mã nguồn được kiểm tra bởi các bài kiểm tra tự động. Đảm bảo rằng tất cả các nhánh (các điều kiện if-else, switch-case) trong mã nguồn được kiểm tra để phát hiện lỗi và đảm bảo chất lượng phần mềm.

Branch coverage cao cho thấy rằng các điều kiện và nhánh trong mã đã được kiểm tra, giúp phát hiện các lỗi tiềm ẩn trong các điều kiện logic.

$$\text{Branch coverage} = \frac{\text{Branches traversed}}{\text{Total number of branches}}$$

Các công cụ như JaCoCo (Java), Istanbul (JavaScript), và Coverage.py (Python)

Function Coverage

Function coverage là một loại chỉ số trong code coverage, đo lường tỷ lệ các hàm hoặc phương thức trong mã nguồn được kiểm tra bởi các bài kiểm tra tự động. Function coverage được tính bằng cách chia số hàm đã được kiểm tra cho tổng số hàm trong chương trình, sau đó nhân với 100 để ra phần trăm.

Path Coverage

Path coverage là một loại chỉ số trong code coverage, đo lường tỷ lệ các đường dẫn thực thi trong mã nguồn được kiểm tra bởi các bài kiểm tra tự động.

Đảm bảo rằng tất cả các đường dẫn có thể trong luồng điều khiển của chương trình được kiểm tra ít nhất một lần.

Path coverage được tính bằng cách chia số đường dẫn đã được kiểm tra cho tổng số đường dẫn có thể trong chương trình, sau đó nhân với 100 để ra phần trăm.

02

Best Practices

Best practice trong unit testing là những phương pháp và kỹ thuật đã được chứng minh là hiệu quả nhất để đảm bảo chất lượng và độ tin cậy của mã nguồn.

I. Unit test dễ bảo trì

- Việc viết unit test nên được thiết kế sao cho dễ bảo trì, giúp cho các lập trình viên có thể dễ dàng hiểu và sửa đổi chúng trong tương lai. Tránh việc phụ thuộc vào cơ sở dữ liệu hoặc lưu trữ lâu dài, sử dụng mock dependencies để đảm bảo tính chính xác của các test.

II. Kiểm thử happy case và edge case

- Cần kiểm thử không chỉ các trường hợp thành công (happy case) mà còn cả các trường hợp biên (edge case) để đảm bảo rằng ứng dụng có thể xử lý tất cả các tình huống một cách hiệu quả. Điều này giúp nâng cao độ tin cậy và giảm thiểu lỗi trong môi trường thực tế.

III. Tránh test trùng lặp và phụ thuộc vào implementation

- Việc viết test trùng lặp sẽ làm giảm tính hiệu quả và dễ bảo trì của bộ test. Các lập trình viên cần đảm bảo rằng các test không phụ thuộc vào các chi tiết cài đặt cụ thể của mã nguồn, giúp cho các test vẫn còn hiệu lực ngay cả khi cấu trúc mã nguồn thay đổi.