플러터 UI 테스트

Flutter UI Testing Guide

학습목표

- 자동화된 위젯 테스트의 개념과 구현 방법 이해
- Flutter Driver를 활용한 통합 테스트 수행 능력 습득
- Firebase Test Lab을 통한 기기 호환성 테스트 방법 학습

UI 테스트 개요

- Flutter 기반 코드의 사용자 인터페이스 테스트 케이스 작성 방법 습득
- 위젯 테스트와 자동화된 위젯 테스트의 개념 이해
- 지속적 통합 파이프라인을 위한 외부 도구 활용법 학습
- Firebase 테스트 도구를 통한 호환성 테스트 방법 이해

자동화된 위젯 테스트

- 사용자 상호작용을 통합한 향상된 UI 테스트 수행
- FloatingActionButtons, Text, ListView 등 위젯 요소에 대한 테스트 통합
- 전체적인 애플리케이션 결함 감소를 위한 추가적인 테스트 커버리지 제공

counter_test.dart

기본 카운터 앱을 사용한 위젯 테스트 예시 코드입니다.

```dart
import 'package:flutter/material.dart'; // Material 디자인 관련 위젯 import
import 'package:flutter_test/flutter_test.dart'; // Flutter 테스트 관련 import
import 'package:test_widget_app/main.dart'; // 테스트할 앱의 main.dart import

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // WidgetTester는 위젯을 찾고 상호작용하는 데 사용됩니다.

    // 1. 앱을 빌드하고 화면에 렌더링합니다.
    await tester.pumpWidget(const MyApp()); // MyApp 위젯을 렌더링

    // 2. 초기 상태를 검증합니다.
    expect(find.text('0'), findsOneWidget); // '0' 텍스트가 정확히 1개 있는지 확인
    expect(find.text('1'), findsNothing); // '1' 텍스트가 없는지 확인

    // 3. '+' 아이콘을 탭합니다.
    await tester.tap(find.byIcon(Icons.add)); // '+' 아이콘을 찾아 탭
    await tester.pump(); // 화면을 다시 렌더링하여 변경 사항 반영

    // 4. 변경된 상태를 검증합니다.
    expect(find.text('0'), findsNothing); // '0' 텍스트가 사라졌는지 확인
    expect(find.text('1'), findsOneWidget); // '1' 텍스트가 정확히 1개 있는지 확인
  });
}
```

login_test.dart 1/2

로그인 기능에 대한 위젯 테스트 예시입니다.

```dart
import 'package:flutter/material.dart'; // Material 디자인 관련 위젯 import
import 'package:flutter_test/flutter_test.dart'; // Flutter 테스트 관련 import

void main() {
  testWidgets("should allow login", (WidgetTester testWorker) async {
    // WidgetTester는 위젯을 찾고 상호작용하는 데 사용됩니다.

    // 1. 테스트 대상 위젯 찾기
    final testUsername = find.byKey(const ValueKey("testUsername")); // username 필드 찾기
    final testPassword = find.byKey(const ValueKey("testPassword")); // password 필드 찾기
    final testLoginBtn = find.byKey(const ValueKey("testLoginBtn")); // login 버튼 찾기

    // 2. 위젯 렌더링
    await testWorker.pumpWidget(const MaterialApp(home: Home())); // Home 위젯을 렌더링

    // 3. 텍스트 필드에 텍스트 입력
    await testWorker.enterText(testUsername, "username"); // username 필드에 "username" 입력
    await testWorker.enterText(testPassword, "password"); // password 필드에 "password" 입력

    // 4. 버튼 탭
    await testWorker.tap(testLoginBtn); // 로그인 버튼 탭

    // 5. 화면 갱신
    await testWorker.pump(); // 화면 갱신

    // 6. 결과 확인
    expect(find.text("Login credentials supplied"), findsOneWidget); // "Login credentials supplied" 텍스트가 있는지 확인
  });
}
```

login_test.dart 2/2

```dart
// Home 위젯은 예시를 위해 간단하게 작성되었습니다.
class Home extends StatelessWidget {
  const Home({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const TextField(key: ValueKey("testUsername"), decoration: InputDecoration(hintText: "Username")),
            const TextField(key: ValueKey("testPassword"), decoration: InputDecoration(hintText: "Password"), obscureText: true),
            ElevatedButton(
              key: const ValueKey("testLoginBtn"),
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text("Login credentials supplied")),
                );
              },
              child: const Text("Login"),
            ),
          ],
        ),
      ),
    );
  }
}
```

integration_test.dart

Flutter Driver를 사용한 통합 테스트 예시입니다.

```dart
import 'package:flutter/material.dart'; // Material 디자인 관련 위젯 import
import 'package:flutter_driver/driver_extension.dart'; // Flutter Driver 관련 import

void main() {
  // Flutter Driver 확장을 활성화합니다.
  enableFlutterDriverExtension();

  // 앱을 실행합니다.
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Integration Test"),
        ),
        body: MyWidget(),
      ),
    );
  }
}

class MyWidget extends StatelessWidget {
  const MyWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: <Widget>[
        const TextField(
          key: Key("username"),
          decoration: InputDecoration(labelText: "Enter Username"),
        ),
        FloatingActionButton(
          key: const Key("additionButton"),
          onPressed: () {
            // ignore: avoid_print
            print("Button Pressed");
          },
          child: const Icon(Icons.add),
        ),
      ],
    );
  }
}
```

integration_test_driver.dart

통합 테스트를 위한 드라이버 코드입니다.

```dart
import 'package:flutter_driver/flutter_driver.dart'; // Flutter Driver 관련 import
import 'package:test/test.dart'; // 테스트 관련 import

void main() {
  group('Integration Test', () {
    FlutterDriver? driver; // FlutterDriver 인스턴스

    // 모든 테스트가 시작되기 전에 실행됩니다.
    setUpAll(() async {
      driver = await FlutterDriver.connect(); // Flutter Driver에 연결
    });

    // 모든 테스트가 완료된 후 실행됩니다.
    tearDownAll(() async {
      if (driver != null) {
        await driver.close(); // Flutter Driver 연결 종료
      }
    });

    test('Should enter username and press button', () async {
      // 1. 요소 찾기
      final txtUsername = find.byValueKey("username"); // username 필드 찾기
      final btnAddition = find.byValueKey("additionButton"); // addition 버튼 찾기

      // 2. 텍스트 입력 및 버튼 탭
      await driver!.tap(txtUsername); // username 필드 탭
      await driver.enterText("Martha Kent"); // "Martha Kent" 텍스트 입력
      await driver.tap(btnAddition); // addition 버튼 탭

      // 3. 결과 확인
      await driver.waitFor(find.text("Button Pressed")); // "Button Pressed" 텍스트가 나타날 때까지 대기
    });
  });
}
```

통합 테스트는 다음과 같은 구성요소로 이루어집니다

1. 테스트 애플리케이션 설정
    - Flutter Driver 확장 활성화
    - 테스트 대상 위젯 구현
    - 각 요소에 Key 할당
2. 드라이버 설정
    - FlutterDriver 연결
    - 테스트 그룹 정의
    - setUp/tearDown 설정

3. 테스트 케이스 구현
    - 요소 찾기
    - 사용자 동작 시뮬레이션
    - 결과 검증
4. 실행
    - flutter drive 명령어로 테스트 실행
    - 테스트 결과 확인

Firebase Test Lab 활용

Firebase Test Lab을 통한 테스트는 다음과 같은 특징이 있습니다:

- Robo 테스트를 통한 자동화된 기기 호환성 테스트
- Android APK/AAB 파일 기반 테스트 수행
- 실제 기기 및 가상 기기에서의 테스트 지원
- 테스트 과정의 스크린샷과 동영상 기록 제공
- iOS의 경우 XCTest 프레임워크 활용

요약

- Flutter에서의 UI 테스트 방법을 학습했습니다.
- 위젯 테스트, 자동화된 위젯 테스트, 통합 테스트의 개념을 이해했습니다.
- Firebase Test Lab을 활용한 기기 호환성 테스트 방법을 학습했습니다.

END