TESTING IN FLUTTER

unit and integration testing

PREREQUISITES

- I won't cover flutter and dart basics.
  - Check out the e-Portfolio of Dennis Rein for that
- Flutter and Dart installation
  - Check out the installation guide on Github
  - https://github.com/JoschuaGoetz/se_flutter_testing#installation

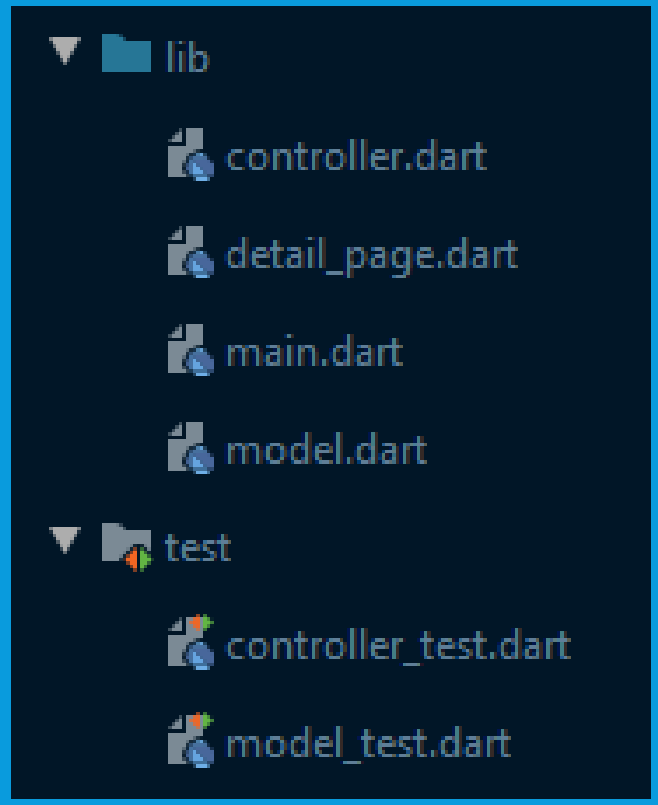
BUILD IN TESTING FRAMEWORK

TYPE	ENVIRONMENT	COVERED HERE	MOCKABLE
Unit	CLI – without UI	yes	yes
Widget	Single Widget in testing environment	no	yes
Integration	Whole app in testing environment	yes	Difficult but yes

UNIT TESTING - STRUCTURE

Folder structure and best practices:

- All tests are in the test directory
- Each test file corresponds to one file in the lib directory.
- Flutter recognizes a file as a test file if it ends with "_test.dart"
- To execute you just have to type "flutter test" from your project root



UNIT TESTING - INSTALLATION

- Add flutter test to your pubspec.yml

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  test: any
```

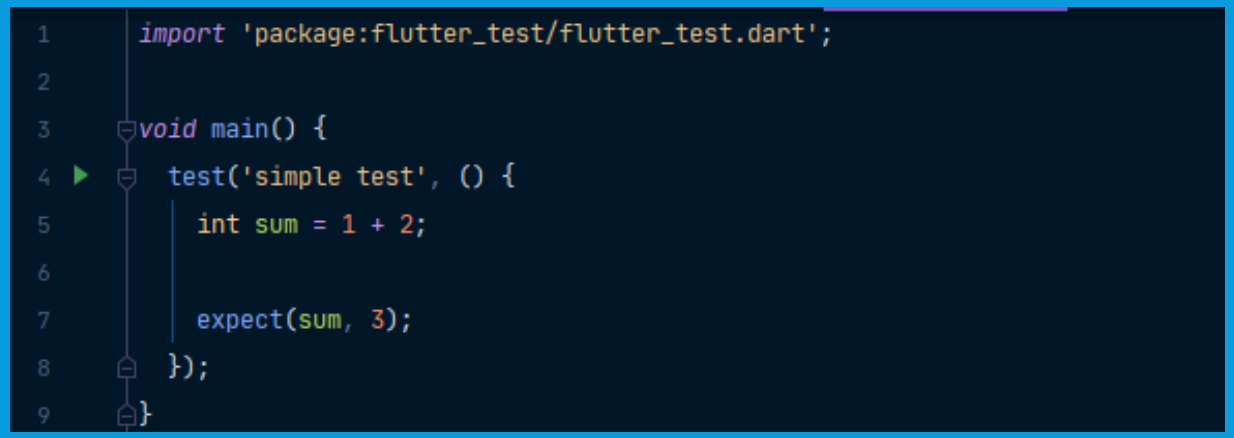
- Run "flutter pub get"

UNIT TESTING - METHODS

- void main ()
  -> Executable file of each test file

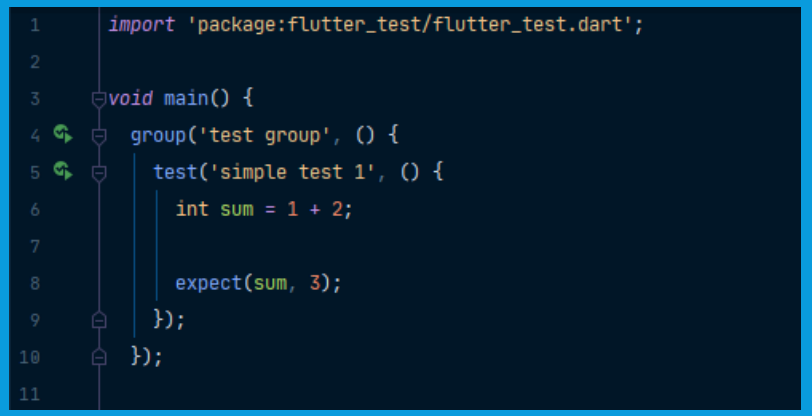
UNIT TESTING - METHODS

- test(description, body)
  - Contains test code
- expect(actual, matcher)
  - Test assertion
  - Must be in a test function
  - The test function only passes if all expects pass



UNIT TESTING - METHODS

- group(description, body)
  - groups test inside



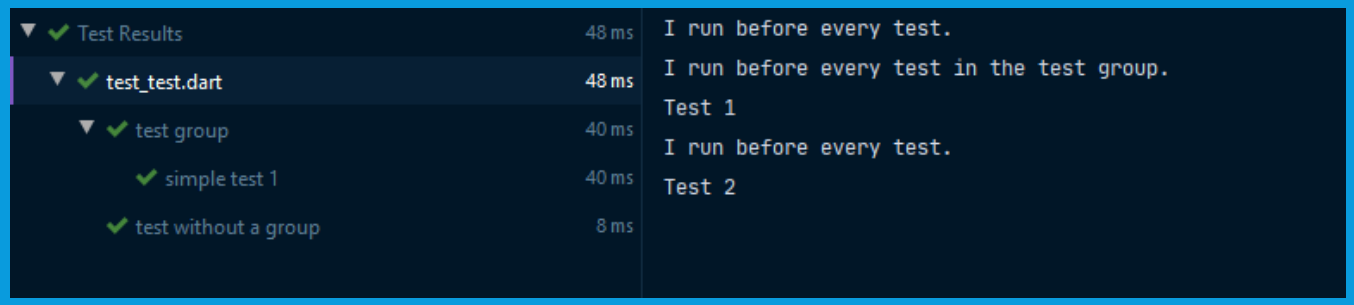
```dart
import 'package:flutter_test/flutter_test.dart';

void main() {
  group('test group', () {
    test('simple test 1', () {
      int sum = 1 + 2;

      expect(sum, 3);
    });
  });

```

UNIT TESTING - METHODS

- setUp(body)
  - runs before every test
  - If in a group, it runs before each test in the group
  - Multiple setUp functions are executed from top to bottom

```dart
import 'package:flutter_test/flutter_test.dart';

void main() {
  setUp(() {
    print('I run before every test.');
  });

  group('test group', () {
    setUp(() {
      print('I run before every test in the test group.');
    });

    test('simple test 1', () {
      print('Test 1');
      int sum = 1 + 2;

      expect(sum, 3);
    });
  });

  test('test without a group', () {
    print('Test 2');
    int sum = 5 + 6;

    expect(sum, 11);
  });
}
```

Console output



UNIT TESTING - METHODS

- setUpAll(body)
  - runs once before any tests are run
  - If in a group, runs once before any tests in the group are run
  - Multiple setUpAll functions are executed from top to bottom

Console output

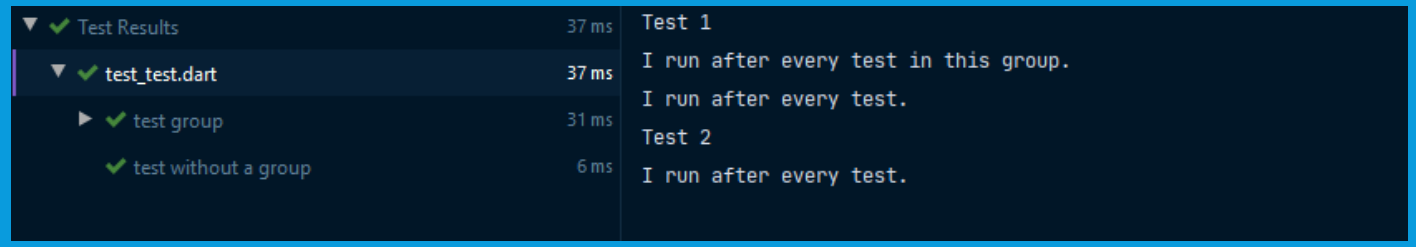
```
$ flutter test test\\test_test.dart
00:02 +0: (setUpAll)
I run once at the start of this test file.
00:02 +0: test group (setUpAll)
I run once before any tests are run in this group.
00:02 +0: test group simple test 1
Test 1
00:02 +1: test group simple test 2
Test 2
00:02 +2: test without a group
Test 3
00:03 +3: All tests passed!
```

```dart
import 'package:flutter_test/flutter_test.dart';

void main() {
  setUpAll(() {
    print('I run once at the start of this test file.');
  });

  group('test group', () {
    setUpAll(() {
      print('I run once before any tests are run in this group.');
    });

    test('simple test 1', () {
      print('Test 1');
      int sum = 1 + 2;

      expect(sum, 3);
    });

    test('simple test 2', () {
      print('Test 2');
      int sum = 3 + 4;

      expect(sum, 7);
    });
  });

  test('test without a group', () {
    print('Test 3');
    int sum = 5 + 6;

    expect(sum, 11);
  });
}
```

UNIT TESTING - METHODS

- tearDown(body)
  - runs after every test
  - If in a group, it runs after each test in the group
  - Multiple tearDown functions are executed from bottom to top

Console output




```dart
import 'package:flutter_test/flutter_test.dart';

void main() {
  tearDown(() {
    print('I run after every test.');
  });

  group('test group', () {
    tearDown(() {
      print('I run after every test in this group.');
    });

    test('simple test 1', () {
      print('Test 1');
      int sum = 1 + 2;

      expect(sum, 3);
    });
  });

  test('test without a group', () {
    print('Test 2');
    int sum = 5 + 6;

    expect(sum, 11);
  });
}
```

UNIT TESTING - METHODS

- tearDownAll(body)
  - runs once after all tests are run
  - If in a group, runs once after all tests in the group are run
  - Multiple tearDownAll functions are executed from top to bottom

Console output

```
$ flutter test test\\test_test.dart
00:03 +0: test group simple test 1
Test 1
00:03 +1: test group simple test 2
Test 2
00:03 +2: test group (tearDownAll)
I run once after all tests are run in this group.
00:03 +2: test without a group
Test 3
00:03 +3: (tearDownAll)
I run once at the end of this test file.
00:03 +3: All tests passed!
```

```dart
import 'package:flutter_test/flutter_test.dart';

void main() {
  tearDownAll(() {
    print('I run once at the end of this test file.');
  });

  group('test group', () {
    tearDownAll(() {
      print('I run once after all tests are run in this group.');
    });

    test('simple test 1', () {
      print('Test 1');
      int sum = 1 + 2;

      expect(sum, 3);
    });

    test('simple test 2', () {
      print('Test 2');
      int sum = 3 + 4;

      expect(sum, 7);
    });
  });

  test('test without a group', () {
    print('Test 3');
    int sum = 5 + 6;

    expect(sum, 11);
  });
}
```

UNIT TESTING - COVERAGE

- To get code coverage just execute yout tests with the "--coverage" flag

```
$ flutter test --coverage
```

- This generates a coverage file called lcov.info
- Flutter has no native representation for the lcov.info
- You need additional tools to make it readable
  - You could use codacy for example

UNIT TESTING – ADDITIONAL RESOURCES

- Additional info if you are interested
  - https://flutter.dev/docs/cookbook/testing/unit
  - https://pub.dev/packages/test
  - https://pub.dev/packages/mockito

INTEGRATION TESTING - STRUCTURE

Folder structure and best practices:

- All files are in the test_driver directory
- You need two files.
  - The executable file.
    - Can have any name. In this case "app.dart"
  - The test file.
    - Name must be "<Name of executable>_test.dart". In this case "app_test.dart"
- Executable with the command - Needs a running android device (e.g emulator)

```
flutter drive --target=test_driver/app.dart
```

- Adjust the name of your executable file

INTEGRATION TESTING - INSTALLATION

- Add flutter driver to your pubspec.yml

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_driver:
    sdk: flutter
  test: any
```

- Run "flutter pub get"

INTEGRATION TESTING – EXECUTABLE FILE

```dart
import 'package:flutter_driver/driver_extension.dart';
import 'package:sepokedex/main.dart' as app;

void main() {
  // Enable the flutter driver extension to run and capture tests
  enableFlutterDriverExtension();

  // Configure things here before the app is launched.

  // Execute the main app
  app.main();
}
```

This is the minimal execution file for each integration test

INTEGRATION TESTING – TEST FILE

- Same structure as a unit test
- The setUpAll and tearDownAll functions are required to ensure correct execution
- The driver is executing your app during the test

```dart
import 'package:flutter_driver/flutter_driver.dart';
import 'package:test/test.dart';

void main() {
  group('starting the app', () {
    FlutterDriver driver;

    // Connect to the Flutter driver before running any tests.
    setUpAll(() async {
      driver = await FlutterDriver.connect();
    });

    // Close the connection to the driver after the tests have completed.
    tearDownAll(() async {
      if (driver != null) {
        driver.close();
      }
    });

    test('you can see pokemon', ...);
  });
}
```

INTEGRATION TESTING - METHODS

Finding widgets:

- With the built-in object "finder" of flutter_driver
- Needs widgets to have a set key property
- finder.byValueKey(key)
  - Returns the widget with the given key

```
final pokemonName = find.byValueKey('pokemonName');
```

```
Align(
  alignment: Alignment.centerLeft,
  child: Text(
    pokemon.name,
    key: Key('pokemonName'),
    style:
        Theme.of(context).textTheme.headline1.copyWith(fontSize: 72),
  ), // Text
), // Align
```

INTEGRATION TESTING - METHODS

Interacting with a widget:

- driver.tap(widget)
  - The driver taps on the given widget
- driver.enterText(text)
  - Tries to enter text into an active text field
- driver.drag(widget, Offset)
  - Drags the given widget based on the given Offset
  - Offset takes two parameters
    - Vertical distance
    - Horizontal distance
    - e.g Offset(500.0, -250.0)

INTEGRATION TESTING - METHODS

Reading from a widget:

- driver.getText(widget)
  - Gets the value of the text property of the given widget

```
String text = await driver.getText(pokemonName);
```

INTEGRATION TESTING – ADDITIONAL RESOURCES

- Additional info if you are interested
  - https://flutter.dev/docs/cookbook/testing/widget/introduction
  - https://pub.dev/packages/mockito
  - https://medium.com/stuart-engineering/mocking-integration-tests-with-flutter-af3b6ba846c7
  - https://flutter.dev/docs/cookbook/testing/integration
  - https://api.flutter.dev/flutter/flutter_driver/FlutterDriver-class.html

DEMO

FLUTTER TESTING

Thank you