



# 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
  - [Link](#)

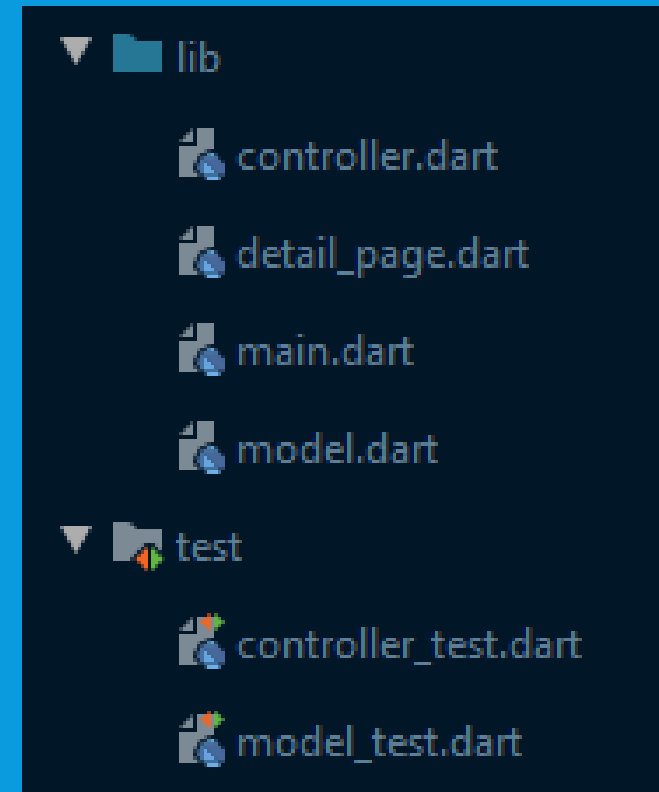
# BUILD IN TESTING FRAMEWORK

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

```
1  void main() {  
2      // Test code here  
3  }
```

# 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

```
1  import 'package:flutter_test/flutter_test.dart';
2
3  void main() {
4    test('simple test', () {
5      int sum = 1 + 2;
6
7      expect(sum, 3);
8    });
9  }
```

# UNIT TESTING - METHODS

- group(description, body)
- groups test inside

```
1  import 'package:flutter_test/flutter_test.dart';
2
3  void main() {
4    group('test group', () {
5      test('simple test 1', () {
6        int sum = 1 + 2;
7
8        expect(sum, 3);
9      });
10   });
11 }
```



# 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

## Console output

▼ ✓ Test Results	48 ms	I run before every test.
▼ ✓ test_test.dart	48 ms	I run before every test in the test group.
▼ ✓ test group	40 ms	Test 1
✓ simple test 1	40 ms	I run before every test.
✓ test without a group	8 ms	Test 2

```
1  import 'package:flutter_test/flutter_test.dart';
2
3  void main() {
4    setUp(() {
5      print('I run before every test.');
```

```
6    });
7
8    group('test group', () {
9      setUp(() {
10        print('I run before every test in the test group.');
```

```
11      });
12
13      test('simple test 1', () {
14        print('Test 1');
```

```
15        int sum = 1 + 2;
16
17        expect(sum, 3);
18      });
19    });
20
21    test('test without a group', () {
22      print('Test 2');
```

```
23      int sum = 5 + 6;
24
25      expect(sum, 11);
26    });
27  }
```

# 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!
```

```
1 import 'package:flutter_test/flutter_test.dart';
2
3 void main() {
4   setUpAll(() {
5     print('I run once at the start of this test file.');
```

```
6   });
7
8   group('test group', () {
9     setUpAll(() {
10      print('I run once before any tests are run in this group.');
```

```
11    });
12
13    test('simple test 1', () {
14      print('Test 1');
```

```
15      int sum = 1 + 2;
```

```
16
17      expect(sum, 3);
18    });
19
20    test('simple test 2', () {
21      print('Test 2');
```

```
22      int sum = 3 + 4;
```

```
23
24      expect(sum, 7);
25    });
26  });
27
28  test('test without a group', () {
29    print('Test 3');
```

```
30    int sum = 5 + 6;
```

```
31
32    expect(sum, 11);
33  });
34 }
```

# 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

▼ ✓ Test Results	37 ms	Test 1
▼ ✓ test_test.dart	37 ms	I run after every test in this group.
▶ ✓ test group	31 ms	I run after every test.
✓ test without a group	6 ms	Test 2
		I run after every test.

```
1  import 'package:flutter_test/flutter_test.dart';
2
3  void main() {
4    tearDown(() {
5      print('I run after every test.');
```

```
6    });
7
8    group('test group', () {
9      tearDown(() {
10        print('I run after every test in this group.');
```

```
11      });
12
13      test('simple test 1', () {
14        print('Test 1');
```

```
15        int sum = 1 + 2;
16
17        expect(sum, 3);
18      });
19    });
20
21    test('test without a group', () {
22      print('Test 2');
```

```
23      int sum = 5 + 6;
24
25      expect(sum, 11);
26    });
27  }
```

# 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!
```

```
1 import 'package:flutter_test/flutter_test.dart';
2
3 void main() {
4   tearDownAll(() {
5     print('I run once at the end of this test file.');
```

```
6   });
7
8   group('test group', () {
9     tearDownAll(() {
10      print('I run once after all tests are run in this group.');
```

```
11    });
12
13    test('simple test 1', () {
14      print('Test 1');
```

```
15      int sum = 1 + 2;
16
17      expect(sum, 3);
18    });
19
20    test('simple test 2', () {
21      print('Test 2');
```

```
22      int sum = 3 + 4;
23
24      expect(sum, 7);
25    });
26  });
27
28  test('test without a group', () {
29    print('Test 3');
```

```
30    int sum = 5 + 6;
31
32    expect(sum, 11);
33  });
34 }
```

# UNIT TESTING - COVERAGE

- To get code coverage just execute your 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

```
1  import 'package:flutter_driver/driver_extension.dart';
2  import 'package:sepokedex/main.dart' as app;
3
4  void main() {
5      // Enable the flutter driver extension to run and capture tests
6      enableFlutterDriverExtension();
7
8      // Configure things here before the app is launched.
9
10     // Execute the main app
11     app.main();
12 }
```

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

```
1  import 'package:flutter_driver/flutter_driver.dart';
2  import 'package:test/test.dart';
3
4  void main() {
5    group('starting the app', () {
6      FlutterDriver driver;
7
8      // Connect to the Flutter driver before running any tests.
9      setUpAll(() async {
10        driver = await FlutterDriver.connect();
11      });
12
13      // Close the connection to the driver after the tests have completed.
14      tearDownAll(() async {
15        if (driver != null) {
16          driver.close();
17        }
18      });
19
20      test('you can see pokemon', ...);
41    });
42  }
```

# 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](https://api.flutter.dev/flutter/flutter_driver/FlutterDriver-class.html)

DEMO

FLUTTER TESTING

Thank you