ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Flutter Mobile Developer

Тема 22. UI- и Unit-тестирование

Цель урока

- продолжить работать над приложением (на этом этапе напишем юнит тесты на наши сервисы)

Цель урока

- после занятия вы сможете:
  - писать Unit-тесты;
  - писать UI-тесты;
  - писать интеграционные тесты;
  - и даже больше! Golden Tests

О чем будем говорить

- Unit tests
- Widget tests
- Integration tests
- Code coverage

Для чего нужны юнит тесты ?

Для чего нужны юнит тесты ?

Вы написали код. Как теперь защитить логику вашего кода от изменений, которые могут привнести другие разработчики?

Для чего нужны юнит тесты ?

Вы написали код. Как теперь защитить логику вашего кода от изменений, которые могут привнести другие разработчики?

Защитите его, покрыв юнит тестами!

Для чего нужны юнит тесты ?

- Unit тесты пишутся на юниты - минимальным юнитом является функция, метод класса.
- Но есть сложившаяся практика минимальным юнитом считать класс и покрытие делать целиком на класс.

Unit Tests - Pure Dart SDK

DEMO

Unit Tests - pubspec dev dependency

```
dev_dependencies:
  test: ^1.15.7
```

Unit Tests

test 1.15.7

Published Nov 23, 2020 • dart.dev • Latest: 1.15.7 / Prerelease: 1.16.0-nullsafety.13

DART | NATIVE | JS | FLUTTER | ANDROID | IOS | WEB

98

Readme | Changelog | Installing | Versions | Scores

`test` provides a standard way of writing and running tests in Dart.

- Writing Tests
- Running Tests
  - Sharding Tests
  - Collecting Code Coverage
  - Restricting Tests to Certain Platforms
  - Platform Selectors
  - Running Tests on Node.js
- Asynchronous Tests
  - Stream Matchers
- Running Tests With Custom HTML

98 LIKES | 80 PUB POINTS | 98% POPULARITY

Publisher

dart.dev

Metadata

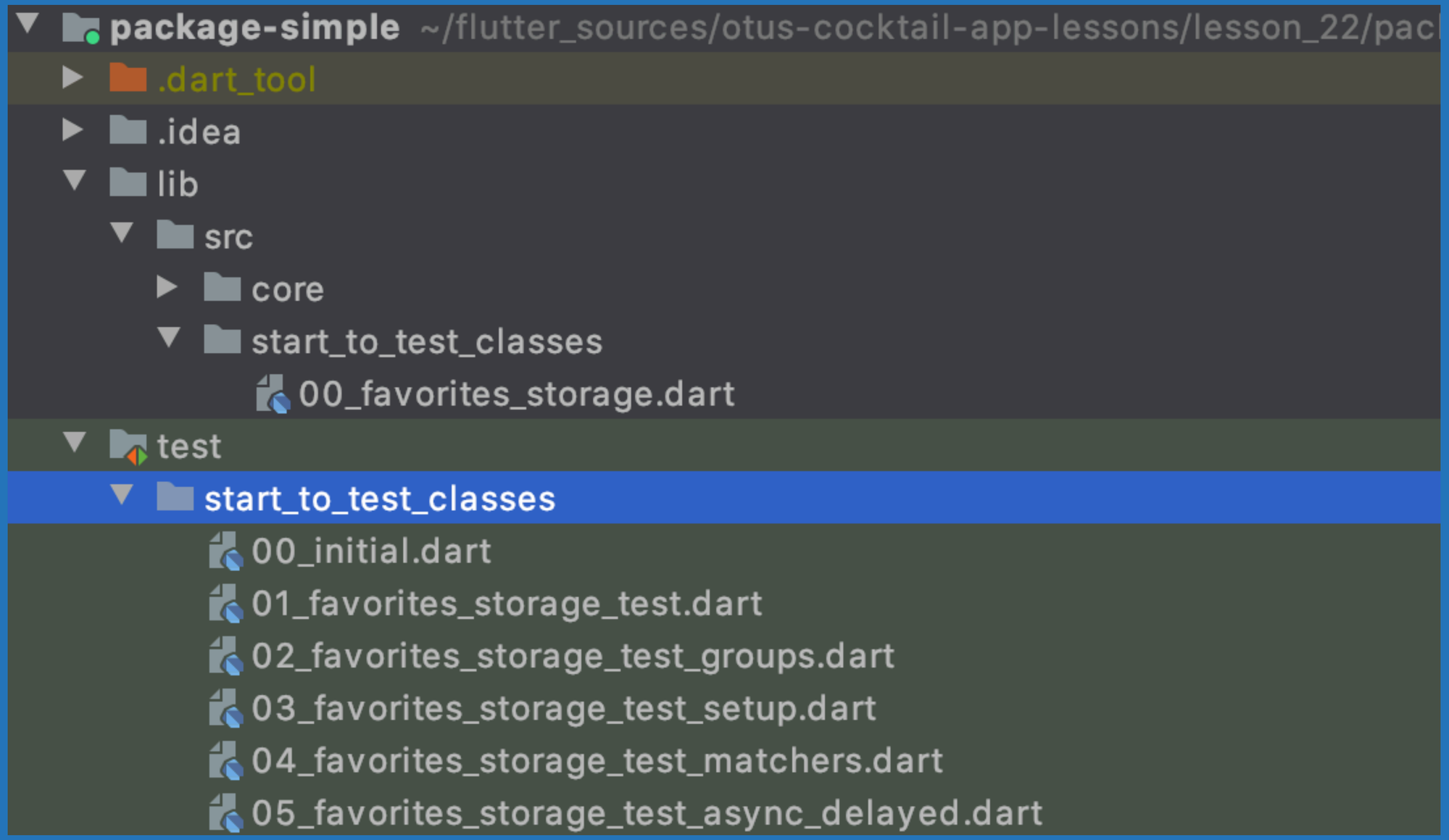
A full featured library for writing and running Dart tests.

Homepage

Repository (GitHub)

View/report issues

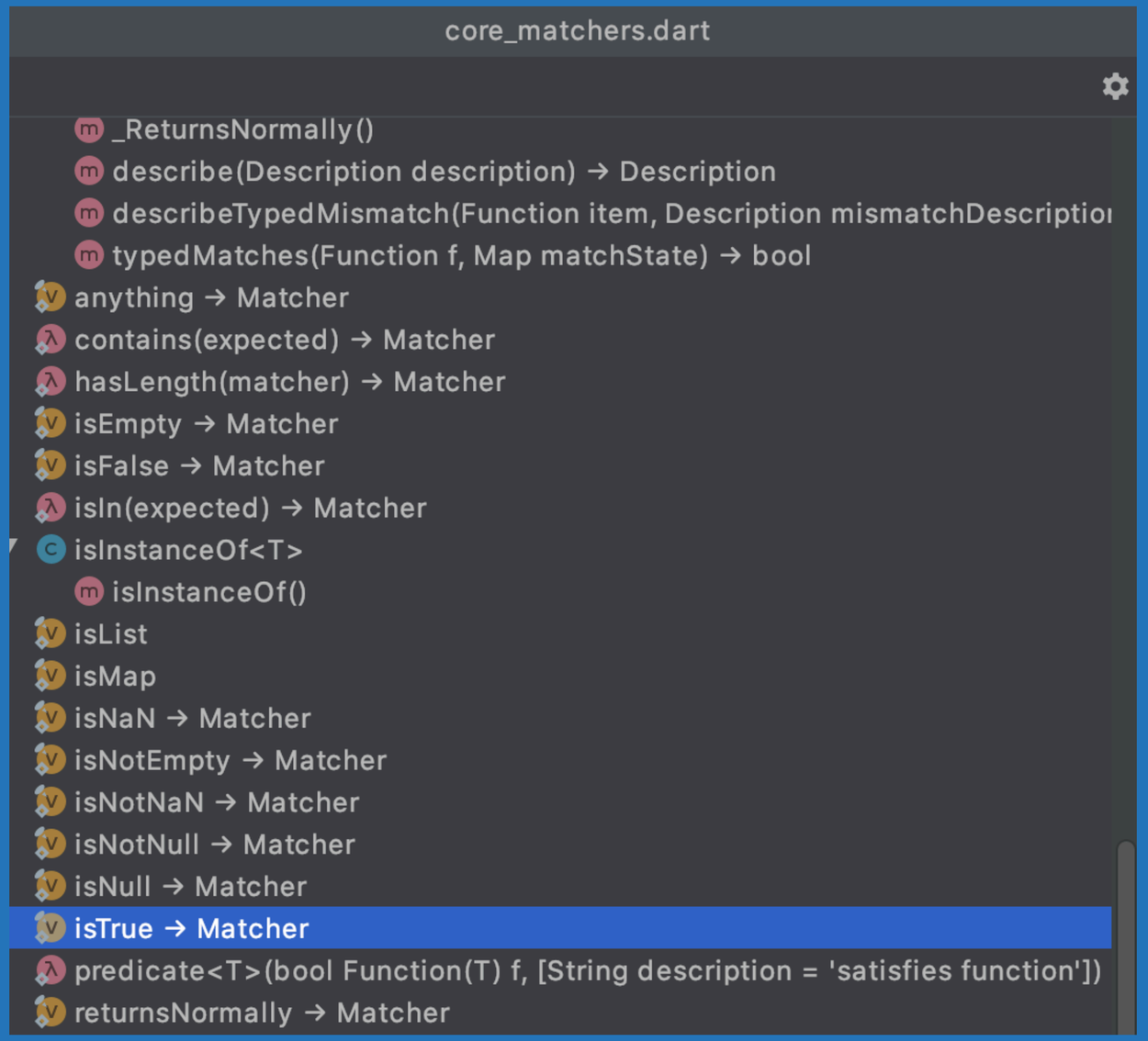
Unit Tests - Folders structure



Unit Tests - Unit Test File Structure

```dart
void main() {
  setUpAll(() => print('setUpAll callback'));

  setUp(() => print('setUp callback'));

  test('some description', () {
    print('we are here!');
  });

  tearDown(() => print('tearDown callback'));

  tearDownAll(() => print('tearDownAll callback'));
}
```

Unit Tests - matchers



Unit Tests - matchers

```dart
/// The base class for all matchers.
///
/// [matches] and [describe] must be implemented by subclasses.
///
/// Subclasses can override [describeMismatch] if a more specific description is
/// required when the matcher fails.
abstract class Matcher {
  const Matcher();

  /// Does the matching of the actual vs expected values.
  ///
  /// [item] is the actual value. [matchState] can be supplied
  /// and may be used to add details about the mismatch that are too
  /// costly to determine in [describeMismatch].
  bool matches(item, Map matchState);

  /// Builds a textual description of the matcher.
  Description describe(Description description);

  /// Builds a textual description of a specific mismatch.
  ///
  /// [item] is the value that was tested by [matches]; [matchState] is
  /// the [Map] that was passed to and supplemented by [matches]
  /// with additional information about the mismatch, and [mismatchDescription]
  /// is the [Description] that is being built to describe the mismatch.
  ///
```

Unit Tests - common core matchers

```dart
void main() {
  final favoritesStorage = FavoritesStorage();

  group('Favorites storage should', () {
    test('be empty in initial state', () {
      expect(favoritesStorage.isEmpty, isTrue);
    });

    test('not be empty after adding the cocktail definition', () {
      const expectedId = 'expected id';
      final cocktailDefinition = CocktailDefinition(id: expectedId, name: null, drinkThumbUrl: null, isFavourite: null);
      favoritesStorage.add(cocktailDefinition);

      expect(favoritesStorage.isEmpty, isFalse);
    });
  });
}
```

Unit Tests - custom matchers

```
27
28    test('not contain cocktail definition if it has not been placed to storage', () {
29      expect(favoritesStorage.contains(cocktailDefinition1), isFalse);
30
31      favoritesStorage.add(cocktailDefinition1);
32      favoritesStorage.add(cocktailDefinition2);
33      expect(favoritesStorage.getAll().last, CocktailDefinitionMatcher(cocktailDefinition1));
34    });
35  });
36  }
37
38  class CocktailDefinitionMatcher extends Matcher {
39    final CocktailDefinition _expected;
```

Run: tests in 04_favorites_storage_test_ma...

Tests failed: 1, passed: 1 of 2 tests – 165 ms

Test Results	165 ms
<no name>	165 ms
Favorites storage should	165 ms
contain cocktail definition after its stori	79 ms
not contain cocktail definition if it has n	86 ms

```
package:test_api                                                    expect
test/start_to_test_classes/04_favorites_storage_test_matchers.dart 33:7  main.<fn>.<fn>

Expected: <Instance of 'CocktailDefinition'>
  Actual: <Instance of 'CocktailDefinition'>
   Which:
           CocktailDefinition.cocktail definition id to be 'expected id 1', but 'expected id 2' found;
```

Unit Tests - custom matchers

```
class CocktailDefinitionMatcher extends Matcher {
  final CocktailDefinition _expected;
  final List<String> _mismatchDescriptionList = [];

  CocktailDefinitionMatcher(this._expected) : super();

  @override
  Description describe(Description description) => description.addDescriptionOf(_expected);

  @override
  Description describeMismatch(dynamic item, Description mismatchDescription, Map matchState, bool verbose) =>
      mismatchDescription.addDescriptionOf(_expected).replace(
            _mismatchDescriptionList.join(),
          );

  String formatFieldMatchError(String fieldName, String expectedValue, String actualValue) =>
      '\n CocktailDefinition.$fieldName to be \'$expectedValue\', but \'$actualValue\' found;';

  @override
  bool matches(dynamic item, Map matchState) {
    if (item is! CocktailDefinition) {
      return false;
    }

    if (item.id != _expected.id) {
      _mismatchDescriptionList.add(formatFieldMatchError('cocktail definition id', '${_expected.id}', '${item.id}'));
    }
```

Unit Tests - FakeAsync

fake_async 1.1.0

Published Apr 6, 2020 • Latest: 1.1.0 / Prerelease: 1.2.0-nullsafety.3

DART | NATIVE | JS | FLUTTER | ANDROID | IOS | WEB

11

Readme | Changelog | Installing | Versions | Scores

This package provides a `FakeAsync` class, which makes it easy to deterministically test code that uses asynchronous features like `Future`s, `Stream`s, `Timer`s, and microtasks. It creates an environment in which the user can explicitly control Dart's notion of the "current time". When the time is advanced, `FakeAsync` fires all asynchronous events that are scheduled for that time period without actually needing the test to wait for real time to elapse.

For example:

```
import 'dart:async';

import 'package:fake_async/fake_async.dart';
import 'package:test/test.dart';

void main() {
  test("Future.timeout() throws an error once the timeout is up", () {
```

11 LIKES | 100 PUB POINTS | 96% POPULARITY

Metadata

Fake asynchronous events such as timers and microtasks for deterministic testing.

Repository (GitHub)

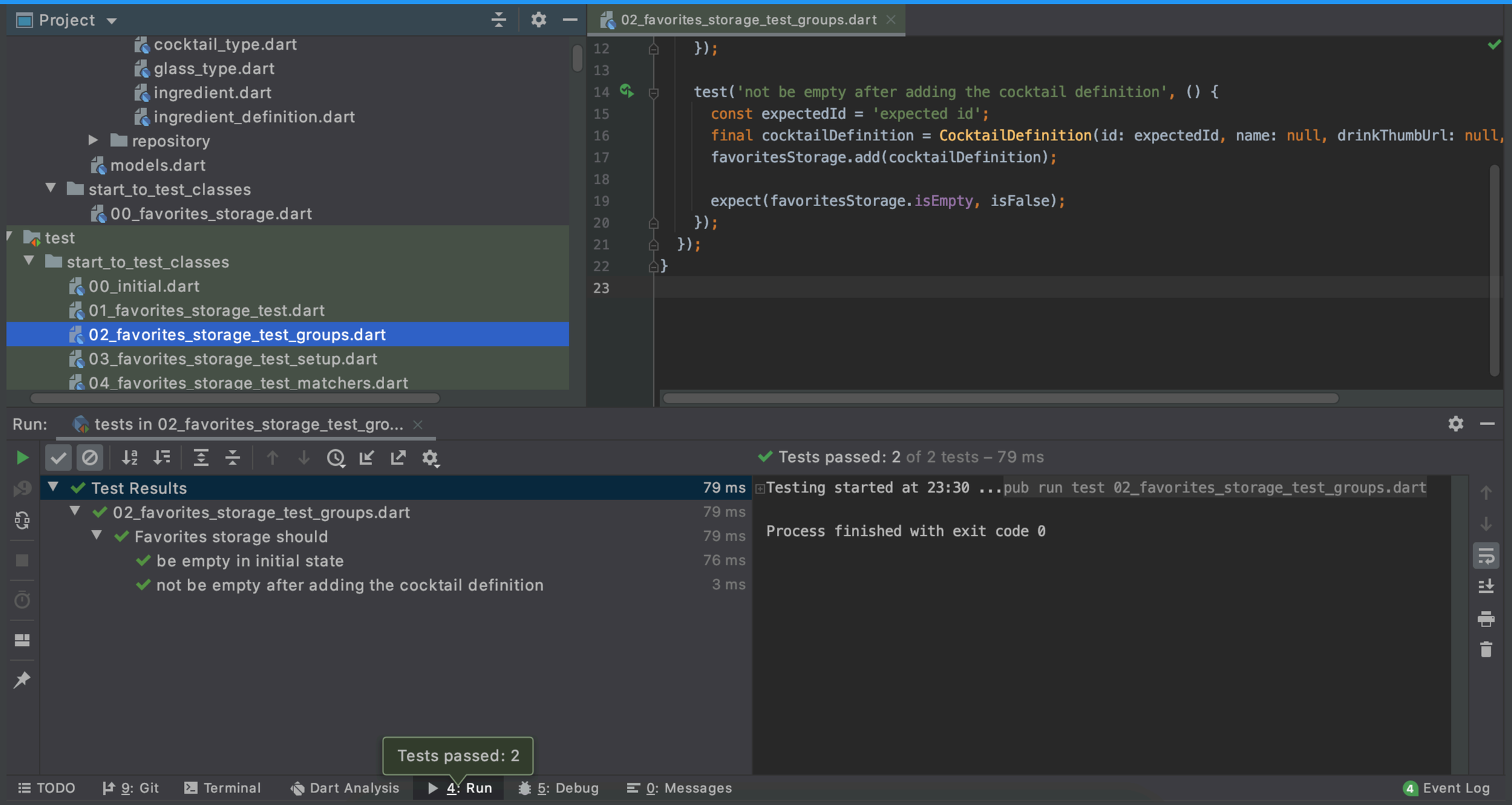
View/report issues

Documentation

API reference

Uploaders

Unit Tests - running the tests in IDE



Unit Tests - Running the tests - pub run test

```
marrrusya@marrrusya-osx1 package-simple % pub run test
00:02 +0: test/start_to_test_classes/00_favorites_storage_test.dart: (setUpAll)

setUpAll callback
00:02 +0: test/start_to_test_classes/00_favorites_storage_test.dart: Favorites storage should be empty in initial state

setUp callback
tearDown callback
00:02 +1: test/start_to_test_classes/00_favorites_storage_test.dart: Favorites storage should not be empty after adding the cocktail
on
setUp callback
tearDown callback
00:02 +2: test/start_to_test_classes/00_favorites_storage_test.dart: (tearDownAll)

tearDownAll callback
00:02 +2: All tests passed!
```

Unit Tests - @TestOn selectors

https://github.com/dart-lang/test/tree/master/pkgs/test#platform-selectors

Platform Selectors

Platform selectors use the boolean selector syntax defined in the `boolean_selector` package, which is a subset of Dart's expression syntax that only supports boolean operations. The following identifiers are defined:

- `vm`: Whether the test is running on the command-line Dart VM.
- `chrome`: Whether the test is running on Google Chrome.
- `phantomjs`: Whether the test is running on PhantomJS.
- `firefox`: Whether the test is running on Mozilla Firefox.
- `safari`: Whether the test is running on Apple Safari.
- `ie`: Whether the test is running on Microsoft Internet Explorer.
- `node`: Whether the test is running on Node.js.
- `dart-vm`: Whether the test is running on the Dart VM in any context. It's identical to `!js`.
- `browser`: Whether the test is running in any browser.
- `js`: Whether the test has been compiled to JS. This is identical to `!dart-vm`.

Мокирование

Мокирование - Mockito

mockito 4.1.3

Published Oct 21, 2020 • dart.dev • Latest: 4.1.3 / Prerelease: 5.0.0-nullsafety.0

DART | NATIVE | JS | FLUTTER | ANDROID | IOS | WEB

226

Readme | Changelog | Example | Installing | Versions | Scores

mockito

Mock library for Dart inspired by Mockito.

pub v4.1.3 | build passing

Let's create mocks

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

// Real class
class Cat {
```

226 LIKES | 80 PUB POINTS | 98% POPULARITY

Publisher

dart.dev

Metadata

A mock framework inspired by Mockito.

Repository (GitHub)

View/report issues

Documentation

API reference

Мокирование - Mockito

```
test('False if http service is answered with Error', () async {
  when(httpServiceMock.get(any)).thenAnswer((realInvocation) => Future.value(HttpResponse.error));
  favoritesStorage = FavoritesStorage(httpServiceMock);

  final cocktailDefinition = CocktailDefinition(id: expectedId, name: null, drinkThumbUrl: null, isFavourite: null);
  final actualOperationResult = await favoritesStorage.add(cocktailDefinition);

  expect(actualOperationResult, isFalse);
  verify(httpServiceMock.get(any)).called(1);
  verify
```

```
verify                    Verification (package:mockito/mockito.dart)
verifyInOrder    _InOrderVerification (package:mockito/mockito.dar…
verifyNever               Verification (package:mockito/mockito.dart)
verifyNoMoreInteractions(va…   void (package:mockito/mockito.dart)
verifyZeroInteractions(var …   void (package:mockito/mockito.dart)
Press ⏎ to insert, ⇥ to replace
```

Все ли понятно по текущей теме?

Ставьте + , если все понятно.

Сообщите голосом в zoom или
напишите в чат, если есть вопросы.

О чем будем говорить

Unit tests

Widget tests

Integration tests

Code coverage

Widget tests

Widget tests - это компонентные тесты, которые позволят вам покрыть логику ваших виджетов в течение жизненного цикла, контроллируемого Flutter.

Цель таких тестов - быть уверенным в том, что ваш виджет взаимодействует и строит свой UI так, как вы это закладываете в его логике.

При этом виджет может создавать как свои дочерние виджеты, так и реагировать на жесты (внешние события).

Widget tests

DEMO

Widget tests

- Во Flutter приложении вы можете использовать и test package и flutter test package
- Для тестирования виджетов вместо функции test() используем testWidgets()
- После получения инстанса WidgetTester выполняем pumpWidget для построения дерева виджетов
- Наследники от класса Finder (byText, byWidget, bySemanticLabel и пр.) позволяют найти необходимые виджеты и применить к ним соотвествующие matchers в логике вашего теста

Widget tests - pubspec dev_dependency

```
dev_dependencies:
  flutter_test:
    sdk: flutter
```

Widget tests

Flutter › flutter_test › WidgetTester class

CLASSES

AccessibilityGuideline
AnimationSheetBuilder
AutomatedTestWidgetsFl...
ChainedFinder
CommonFinders
ComparisonResult
CustomMatcher
CustomMinimumContras...
DefaultTestVariant
DefaultWebGoldenComp...
Description
Evaluation
Finder

pumpFrames(Widget target, Duration maxDuration,
[Duration interval = const Duration(milliseconds: 16, microseconds: 683)]) → Future<void>
Repeatedly pump frames that render the `target` widget with a fixed time `interval` as many as `maxDuration` allows. [...]

pumpWidget(Widget widget, [Duration duration, EnginePhase phase = EnginePhase.sendSemanticsUpdate])
→ Future<void>
Renders the UI from the given `widget`. [...]

renderObject<T extends RenderObject>(Finder finder) → T
The render object of the matching widget in the widget tree. [...]
inherited

renderObjectList<T extends RenderObject>(Finder finder) → Iterable<T>
The render objects of the matching widgets in the widget tree. [...]
inherited

resetTestTextInput() → void
Ensures that testTextInput is registered and TestTextInput.log is reset. [...]

Widget tests - Оберните ваш виджет в Material

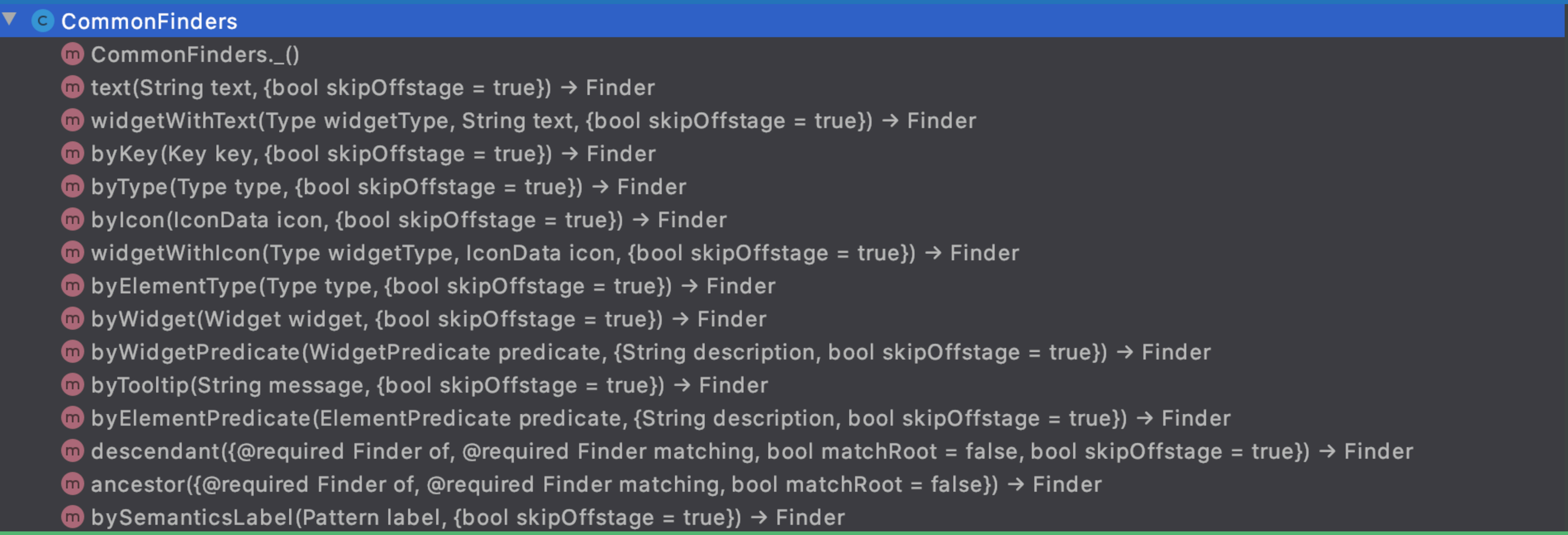
```
Widget _wrap(Widget child) => MaterialApp(
  home: Scaffold(body: child),
);

group('CocktailTitle should', () {
  testWidgets('contain favorite icon button for cocktail from favorites',
(WidgetTester tester) async {
    await tester.pumpWidget(_wrap(CocktailTitle(cocktailTitle: cocktailTitle,
isFavorite: isFavorite)));
```

Widget tests - pumpWidget & pump

- pumpWidget позволяет построить дерево виджетов
- pump позволяет дождаться завершения жизненного цикла в Statefull виджете и завершения всех фреймов (в том числе анимаций) через указанный промежуток времени

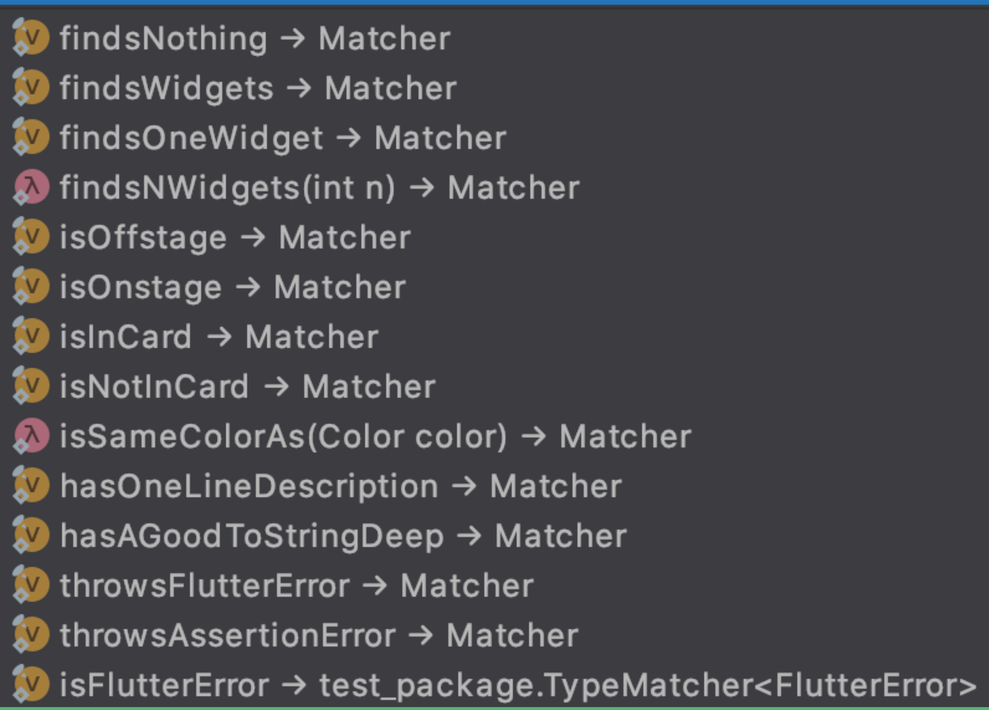
Widget tests - Finder

Используйте наследников класса Finder для анализа дерева виджетов и получения доступа к виджету



Widget tests - Matchers

Используйте наследников класса Matcher из пакета Flutter Test для проверки ожидаемого результата



Widget tests - Matchers

```dart
final titleFinder = find.text(cocktailTitle);
expect(titleFinder, findsOneWidget);

final isFavoriteIconFinder = find.byIcon(Icons.favorite);
expect(isFavoriteIconFinder, findsOneWidget);
```

Widget tests - Semantic Tree

```dart
/// Turns on an overlay that shows the accessibility information
/// reported by the framework.
final bool showSemanticsDebugger;
```

Widget tests - Semantic Tree

```dart
/// A widget that annotates the widget tree with a description of the meaning of
/// the widgets.
///
/// Used by accessibility tools, search engines, and other semantic analysis
/// software to determine the meaning of the application.
///
/// {@youtube 560 315 https://www.youtube.com/watch?v=NvtMt_DtFrQ}
///
/// See also:
///
///  * [MergeSemantics], which marks a subtree as being a single node for
///    accessibility purposes.
///  * [ExcludeSemantics], which excludes a subtree from the semantics tree
///    (which might be useful if it is, e.g., totally decorative and not
///    important to the user).
///  * [RenderObject.describeSemanticsConfiguration], the rendering library API
///    through which the [Semantics] widget is actually implemented.
///  * [SemanticsNode], the object used by the rendering library to represent
///    semantics in the semantics tree.
///  * [SemanticsDebugger], an overlay to help visualize the semantics tree. Can
///    be enabled using [WidgetsApp.showSemanticsDebugger] or
///    [MaterialApp.showSemanticsDebugger].
@immutable
class Semantics extends SingleChildRenderObjectWidget
```

Widget tests - Semantic Tree

```
Widget build(BuildContext context) => Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        Text(
          cocktailTitle ?? '',
          semanticsLabel: ApplicationSemantics.cocktailTitleLabel,
          style: Theme.of(context).textTheme.headline3,
        ),  // Text
        _getIsFavoriteIcon()
      ],
    );  // Row

Widget _getIsFavoriteIcon() => Semantics(
    label: 'Favorite Icon button',
    hint: 'Press to favorite',
    value: '$isFavorite',
    // onTap: () { setState(() { _counter++; }); }
    child: isFavorite
        ? IconButton(
            icon: Icon(Icons.favorite, color: Colors.white),
            onPressed: () {},
          )  // IconButton
        : IconButton(
            icon: Icon(Icons.favorite_border, color: Colors.white),
            onPressed: () {},
```

Widget tests - find bySemanticsLabel

```dart
final titleFinderBySemantic =
find.bySemanticsLabel(ApplicationSemantics.cocktailTitleLabel);
expect(titleFinderBySemantic, findsOneWidget);
```

Все ли понятно по текущей теме?

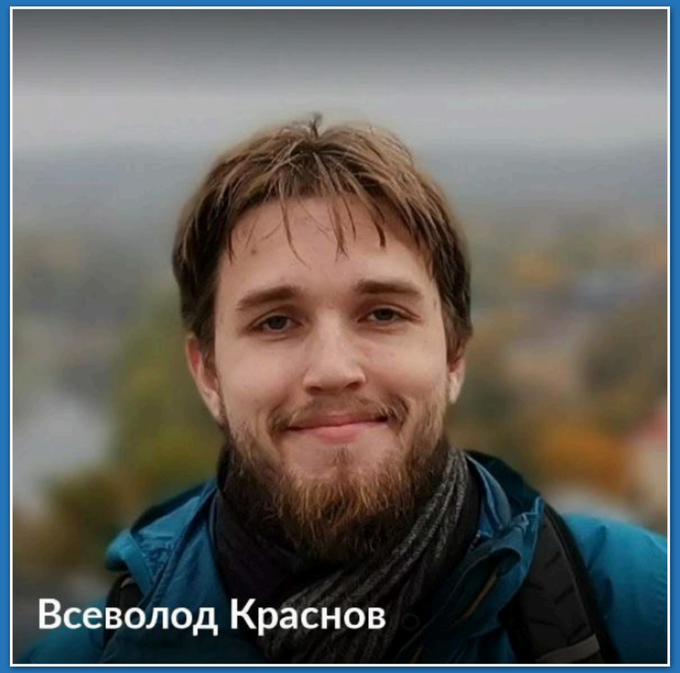
Ставьте + , если все понятно.

Сообщите голосом в zoom или
напишите в чат, если есть вопросы.

Все еще говорим про widget tests

Golden Tests

Golden Tests



О чем будем говорить

Unit tests

Widget tests

Integration tests

Code coverage

Integration tests

Интеграционные тесты - это полноценный запуск вашего приложения на симуляторе или реальном устройстве с целью проверки того факта, что и UI и вся внутренняя логика взаимодействуют корректно и так, как задумывалось.

Здесь нет моков и выполняется реальный application ready код (то есть без моков).

Интеграционные тесты также можно использовать для профилирования производительности.

Integration tests

DEMO

Integration tests - pubspec dev dependency

```
dev_dependencies:
  flutter_driver:
    sdk: flutter
```

Integration tests - Настройка интеграционного места

В случае интеграционного места есть два entry point :

- так называемая instrumented версия приложения (runApp()),
- и код, который «драйвит» первую entry point - собственно, код нашего места.

Integration tests - Настройка интеграционного теста

Но чтобы из кода самого теста мы могли проверять функциональность приложения, запущенного на эмуляторе (или реальном устройстве), нам нужна коммуникация с ним.

Это делается через вызов функции:

`enableFlutterDriverExtension()`

которая запускает VM Service.

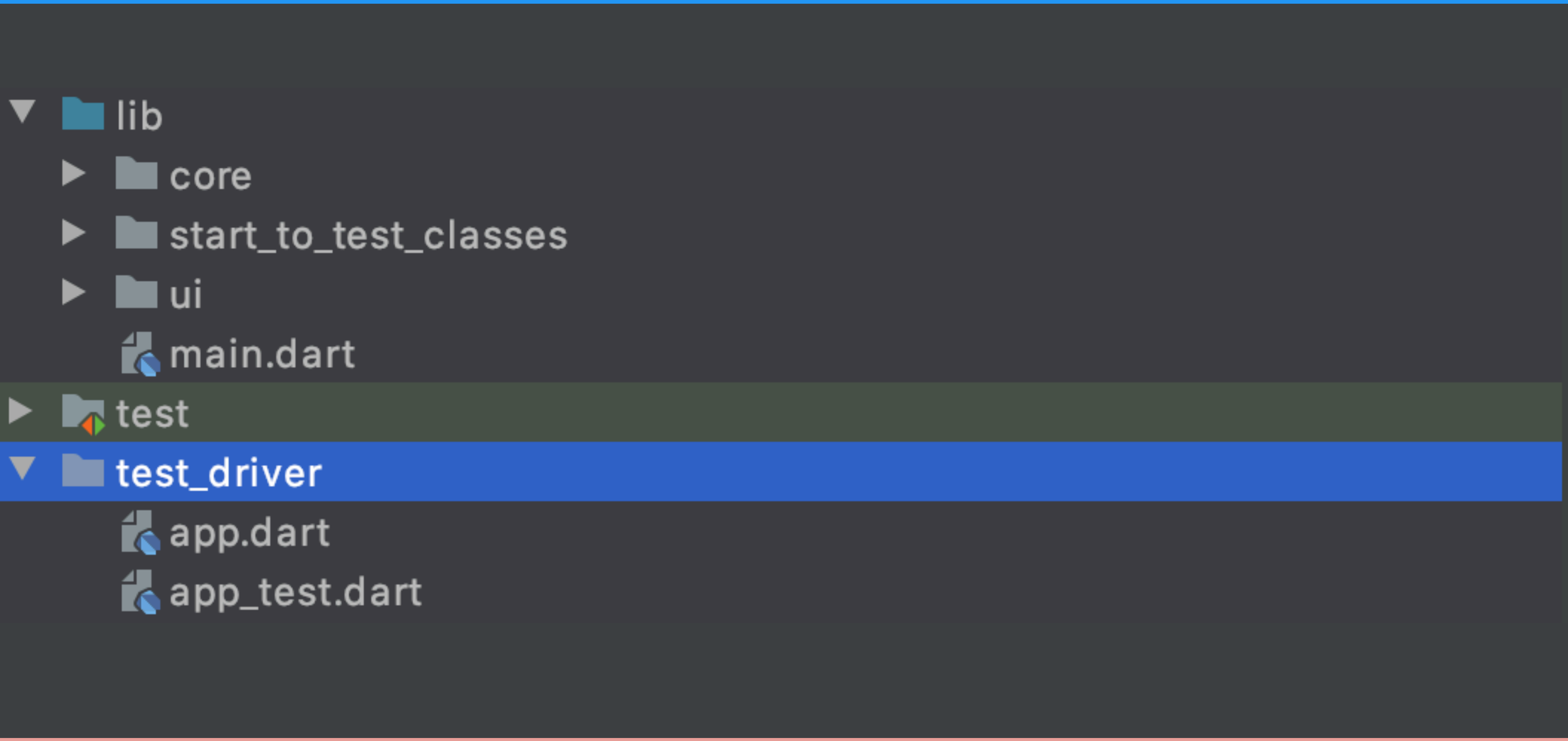
Integration tests - Instrumented App version

```dart
void main() {

  ///
  /// Это то, что отличает instrumented версию от рабочей
  ///
  /// Enables Flutter Driver VM service extension.
  ///
  /// This extension is required for tests that use `package:flutter_driver` to
  /// drive applications from a separate process. In order to allow the driver
  /// to interact with the application, this method changes the behavior of the
  /// framework in several ways - including keyboard interaction and text
  /// editing. Applications intended for release should never include this
  /// method.
  ///
  /// Call this function prior to running your application, e.g. before you call
  /// `runApp`.
  enableFlutterDriverExtension();

  runApp(CocktailOfDayApp());
```

Integration tests - структура места

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

void main() {
  group('', () {
    FlutterDriver driver;

    setUpAll(() async {
      driver = await FlutterDriver.connect();
    });

    test('your test', () async {
    });

    tearDownAll(() async {
      if (driver != null) {
        driver.close();
      }
    });
  });
}
```

Integration tests - Instrumented App version



Integration tests - Running the test

flutter drive —target=test_driver/app.dart

```
marrrusya@marrrusya-osx1 lesson_22_examples % flutter drive --target=test_driver/app.dart
Using device sdk gphone x86 arm.
Starting application: test_driver/app.dart
Installing build/app/outputs/flutter-apk/app.apk...                 3,8s
Running Gradle task 'assembleDebug'...
Running Gradle task 'assembleDebug'... Done                        10,2s
✓ Built build/app/outputs/flutter-apk/app-debug.apk.
I/flutter ( 6867): Observatory listening on http://127.0.0.1:44015/m4Lm6Ht_keA=/
00:00 +0: Cocktail App should (setUpAll)
```

Все ли понятно по текущей теме?

Ставьте + , если все понятно.

Сообщите голосом в zoom или
напишите в чат, если есть вопросы.

О чем будем говорить

Unit tests

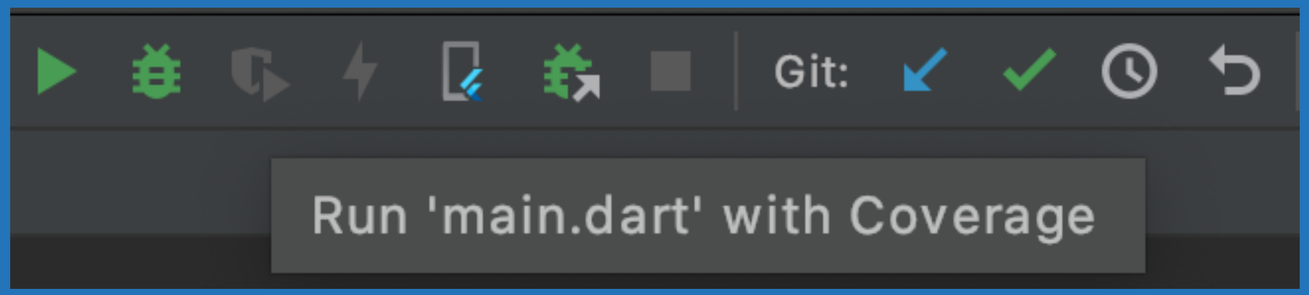
Widget tests

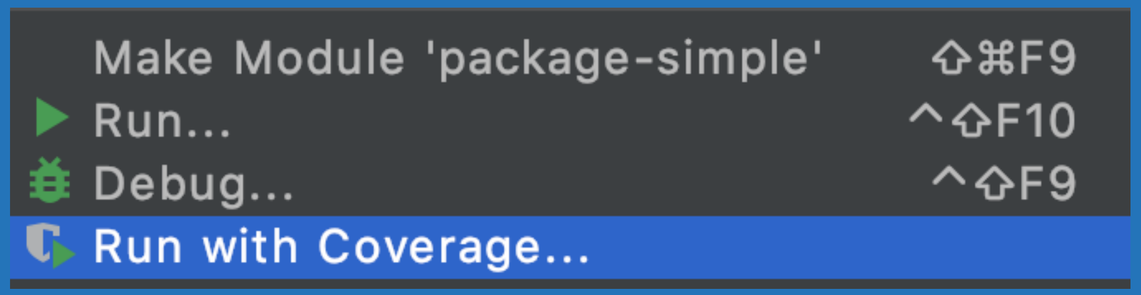
Async

Integration tests

Code coverage

Code coverage





LCOV is a graphical front-end for GCC's coverage testing tool gcov. It collects gcov data for multiple source files and creates HTML pages containing the source code annotated with coverage information. It also adds overview pages for easy navigation within the file structure. LCOV supports statement, function and branch coverage measurement.

Code coverage

flutter test --coverage

если не установлен lcov > brew install lcov

genhtml coverage/lcov.info -o coverage/html

open coverage/html/index.html

Code coverage

LCOV - code coverage report

Current view:	top level		Hit	Total	Coverage
Test:	lcov.info	**Lines:**	17	206	8.3 %
Date:	2020-12-29 04:28:47	**Functions:**	0	0	-

Directory	Line Coverage		Functions	
core/src/dto	0.0 %	0 / 53	-	0 / 0
core/src/model	0.0 %	0 / 33	-	0 / 0
core/src/repository	0.0 %	0 / 90	-	0 / 0
start_to_test_classes	15.4 %	2 / 13	-	0 / 0
ui/pages/details/cocktail_description	88.2 %	15 / 17	-	0 / 0

Generated by: LCOV version 1.15

Code coverage

LCOV - code coverage report

				Hit	Total	Coverage
Current view:	top level - ui/pages/details/cocktail_description - cocktail_title.dart (source / functions)		**Lines:**	15	17	88.2 %
Test:	lcov.info		**Functions:**	0	0	-
Date:	2020-12-29 04:28:47					

Line	Data		Source code
1		:	import 'package:cocktail_app/ui/application_semantics.dart';
2		:	import 'package:flutter/cupertino.dart';
3		:	import 'package:flutter/material.dart';
4		:	
5		:	class CocktailTitle extends StatelessWidget {
6		:	final String cocktailTitle;
7		:	final bool isFavorite;
8		:	
9	1	:	CocktailTitle({this.cocktailTitle, this.isFavorite});
10		:	
11	1	:	@override
12	1	:	Widget build(BuildContext context) => Row(
13		:	crossAxisAlignment: CrossAxisAlignment.center,
14		:	mainAxisAlignment: MainAxisAlignment.spaceBetween,
15	1	:	children: [
16	1	:	Text(
17	1	:	cocktailTitle ?? '',
18		:	semanticsLabel: ApplicationSemantics.cocktailTitleLabel,
19	3	:	style: Theme.of(context).textTheme.headline3,
20		:	),
21	1	:	_getIsFavoriteIcon()
22		:	],
23		:	);
24		:	
25	2	:	Widget _getIsFavoriteIcon() => Semantics(
26		:	label: 'Favorite Icon button',
27		:	hint: 'Press to favorite',
28	2	:	value: '$isFavorite',
29		:	// onTap: () { setState(() { _counter++; }); }
30	1	:	child: isFavorite
31	1	:	? IconButton(
32	1	:	icon: Icon(Icons.favorite, color: Colors.white),
33	0	:	onPressed: () {},
34		:	)

Все ли понятно по текущей теме?

Ставьте + , если все понятно.

Сообщите голосом в zoom или
напишите в чат, если есть вопросы.

Итоги

- Unit tests
- Widget tests
- Async
- Integration tests
- Code coverage

Итоги

	Unit	Widget	Integration
уровень уверенности	низкая	больше	высокая
стоимость сопровождения	низкая	больше	высокая
зависимости	немного	больше	максимально
скорость выполнения	высокая	высокая	медленно

Домашнее задание

Flutter.Тестирование и профилирование. Unit tests

Цель домашнего задания - получить навык работы с юнит тестами и виджет тестами (Unit Testing and Widget Testing)

Пишем юнит тесты на наши сервисы, используемые в работе нашего мобильного приложения.
Пишем тесты на покрытие логики наших виджетов.
Собираем test coverage.

Домашнее задание

- Склонировать соответствующий github репозиторий с заготовкой проекта для этого урока
- Открыть класс сервиса AsyncCocktailRepository, найти код методов, не покрытых юнит-тестами (запустив юнит тесты с code coverage)
- Написать юнит-тесты на метод fetchCocktailDetails (успешное выполнение, выброс exception), используя test (flutter_test) & mockito packages
- Убедиться, что этот метод покрыт юнит тестами - попробуйте собрать test coverage и проанализировать покрытие кода
- Проанализировать класс виджета CocktailDetailPage ((CocktailDescriptionWidget => CocktailTitle => код для отображения признака isFavorite)), написать тест на покрытие кнопки isFavourite - состояние ВИзбранном, Не В Избранном

На усмотрение студента:

- Можно выполнить любую декомпозицию методов AsyncCocktailRepository (рефакторинг - он понадобится, так как используется http)
- Можно вносить изменения в класс CocktailDetailPage (CocktailDescriptionWidget => CocktailTitle => код для отображения признака isFavorite), требуемые для тестирования виджета (семантика)
-

Домашнее задание

Форма сдачи:

- ДЗ Сдается в виде ссылки на pull request на основной репозиторий

Куда сдать ДЗ:

- Отправляется напрямую в ЛК

Куда и кому задавать вопросы, если они возникнут

- По всем вопросам можно обращаться в Slack к студентам, преподавателям в канал группы

Flutter Mobile Developer

А теперь действительно все)

Flutter Mobile Developer

Flutter Mobile Developer


RUSSIA

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Смирнов Андрей wrike

Курс **Мобильная разработка на Flutter**