コードで見る
Flutterアプリの実装

2018-02-09 (Fri)
DroidKaigi 2018
@konifar

Yusuke Konishi

konifar

I'll be back.

Tokyo Japan

yahpeycoy0403@gmail.com

http://konifar.hatenablog.c...

Organizations

Overview | Repositories 95 | Stars 441 | Followers 321 | Following 3

Pinned repositories Order updated.

Customize your pinned repositories

android-material-design-icon-generator-plugin

This plugin help you to set material design icon to your project.

Java ★ 2.1k 214

droidkaigi2016

DroidKaigi 2016 official Android conference app in Tokyo.

Java ★ 676 155

droidkaigi2018-flutter

The unofficial conference app for DroidKaigi 2018 Tokyo

Dart ★ 80 6

fab-transformation

Support Floating Action Button transformation for Android

Java ★ 717 115

kotoha

Kotoha is useful chrome extension that help you to quote a good phrase.

JavaScript ★ 39 3

DroidKaigi/conference-app-2017

The Official Conference App for DroidKaigi 2017 Tokyo

HTML ★ 440 155

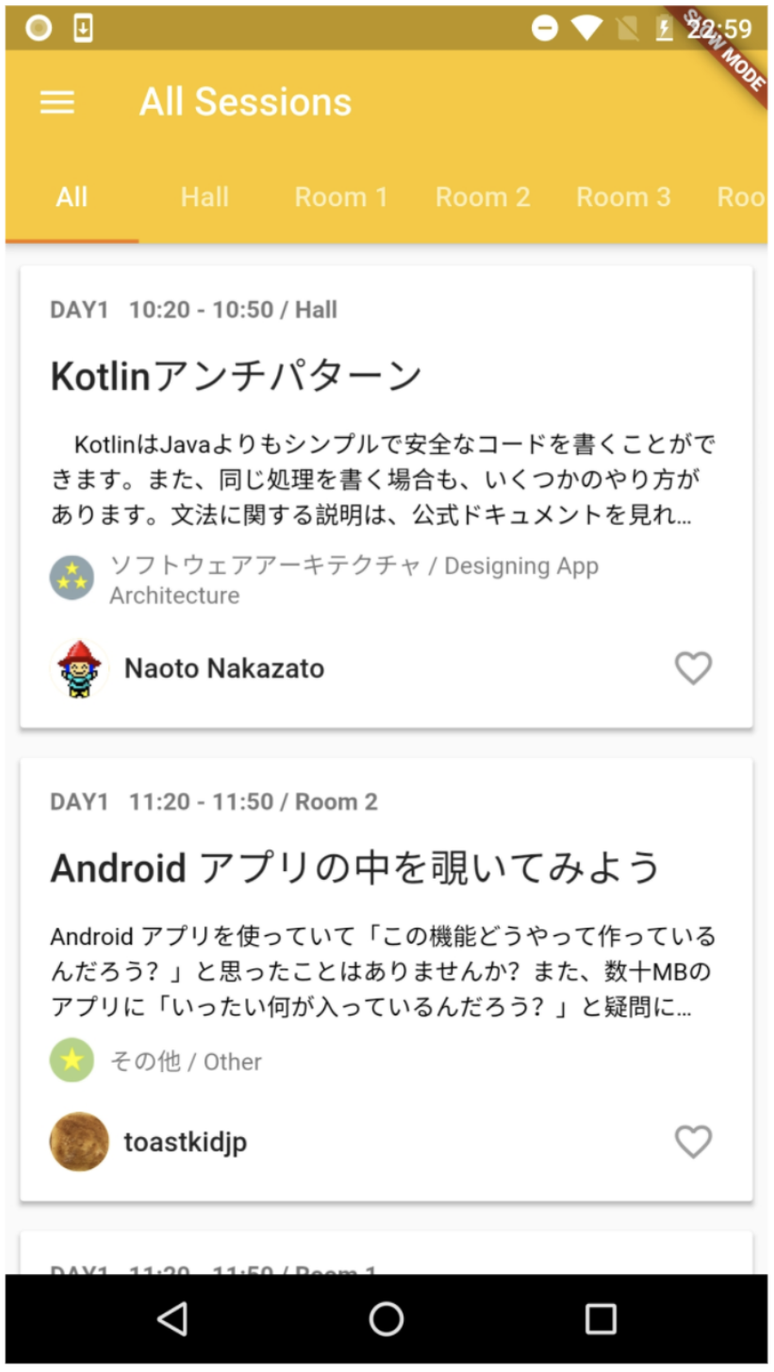
Kyash Inc.

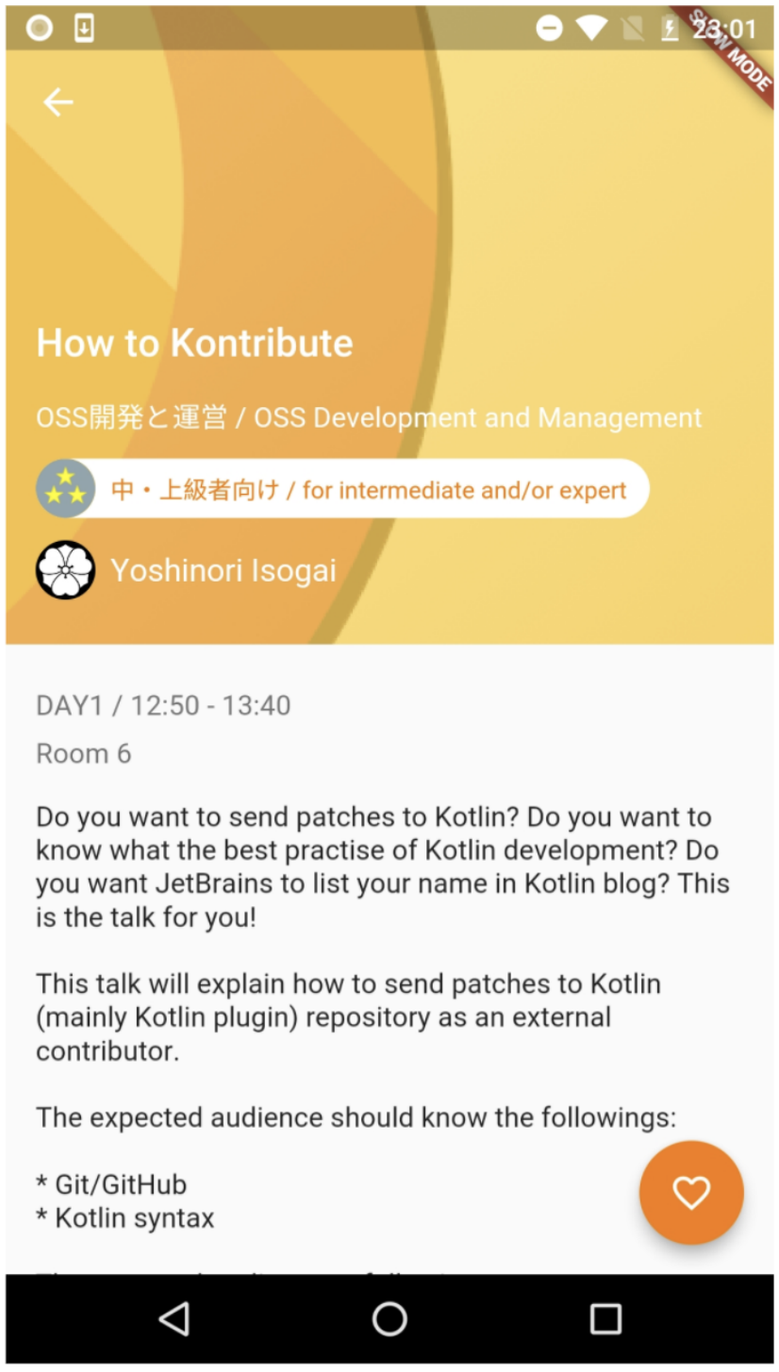
- Sending and receiving money with kyash is easy breeze
- HP : https://kyash.co/
- Store : https://goo.gl/C594Ri

Konifar's QR

Main topic

DroidKaigi2018 Flutter App





https://github.com/konifar/droidkaigi2018-flutter

今日話すこと

- Android/iOSアプリが”**作れる**”のはわかった

- **“業務”で採用できるの？**

業務で採用する時に考えること

- **生存可能性**
  公式のサポート具合、コミュニティの活発さ

- **実装**
  レイアウト、ネットワーク通信、キャッシュ

- **デバッグ、テスト**
  デバッグツール、CI、テストフレームワーク

- **運用**
  Push Notification、 Analytics、テスト配信、リリース

業務で採用する時に考えること

- **生存可能性**
  公式のサポート具合、コミュニティの活発さ

Main topic

- **実装**
  レイアウト、ネットワーク通信、キャッシュ

- **デバッグ、テスト**
  デバッグツール、CI、テストフレームワーク

- **運用**
  Push Notification、Analytics、テスト配信、リリース

業務で採用する時に考えること

- **生存可能性**
  公式のサポート具合、コミュニティの活発さ

- **実装**
  レイアウトのネットワーク通信、キャッシュ

このへんも一部かけ足で触れます

- **デバッグ、テスト**
  デバッグツール、CI、テストフレームワーク

- **運用**
  Push Notification、 Analytics、テスト配信、リリース

Today’s goal

- **Flutter知らなかった人**

  => 「何か作ってみようかな」

- **Flutter知ってた人**

  => 「もしかしたら業務で使えるかも？」

今日話さないこと

- **Dartの言語仕様の話**

  => Javaに似てる。language-tourよい。https://www.dartlang.org/guides/language/language-tour

- **Flutterのセットアップの話**

  => 公式ドキュメントが手厚く説明してくれている。 https://flutter.io/setup/

- **Flutterの内部実装の話**

  => https://docs.google.com/presentation/d/1B3p0kP6NV_XMOimRV09Ms75ymljU5gr6GGIX74Om_DE/edit#slide=id.g2388ebc691_2_112

- **設計の話**

  => ほとんどReactと同じ議論。 http://fluttersamples.com/

- **既存アプリにFlutterを導入する話**

  => 仕組みとしてはできる。 https://github.com/flutter/flutter/tree/master/examples/platform_view

- **iOSの話**

  => リリースは大変とだけ伝えておきたい。

Agenda

1. **Flutterのおさらい (5分)**
    1. Flutterとは
    2. ReactNativeとの違い
    3. Androiderから見たFlutter
2. **Flutterのコード (15分)**
    1. Widget（基本、Widgetツリーの構築）
    2. データの扱い（ネットワーク通信、キャッシュ）
3. **一問一答 (5分)**

1. Flutterのおさらい

Flutterとは

- OS推奨のデザインに合わせた綺麗なアプリを素早く作るためのクロスプラットフォームSDK

- Google製。開発言語はDart

- レイアウトはxmlではなくDartでWidget Treeを書く

ReactNativeとの違い

	Flutter	ReactNative
Language	Dart	Javascript
UI Support	Support officially	Depends on 3rd party libraries
Rendering	Own engine	JS bridge & Native

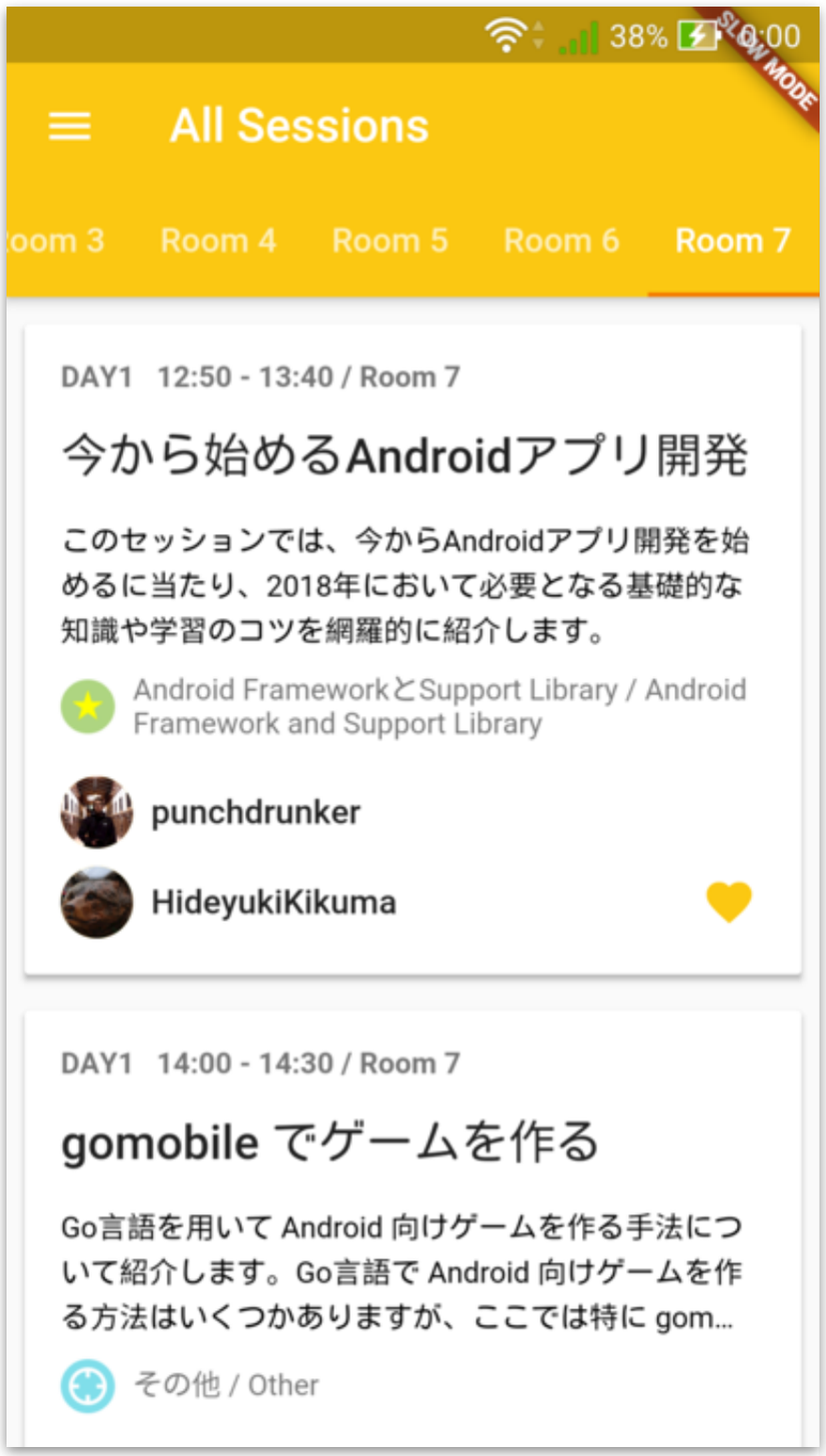
https://flutter.io/faq/

豊富なWidget

- Material Design Guidelineに出てくるパターンのUIはほぼ全て公式にサポートされている

- **Widgetを知り、使いこなすこと**が高速に開発していく上でとても重要

例：Scaffold




```
return new Scaffold(
  appBar: new AppBar(
    title: new Text(_title),
    elevation: _page.hasTab ? 0.0 : 4.0,
  ),
  drawer: new MyDrawer(
    items: _pages,
    currentIndex: _currentIndex,
    onTap: (int index) {
      setState(() {
        _page.controller.reverse();
        _currentIndex = index;
        _page.controller.forward();
        _title = _page.title;
      });
    },
  ),
  body: new Center(
    child: _buildTransitionsStack(),
  ),
);
```

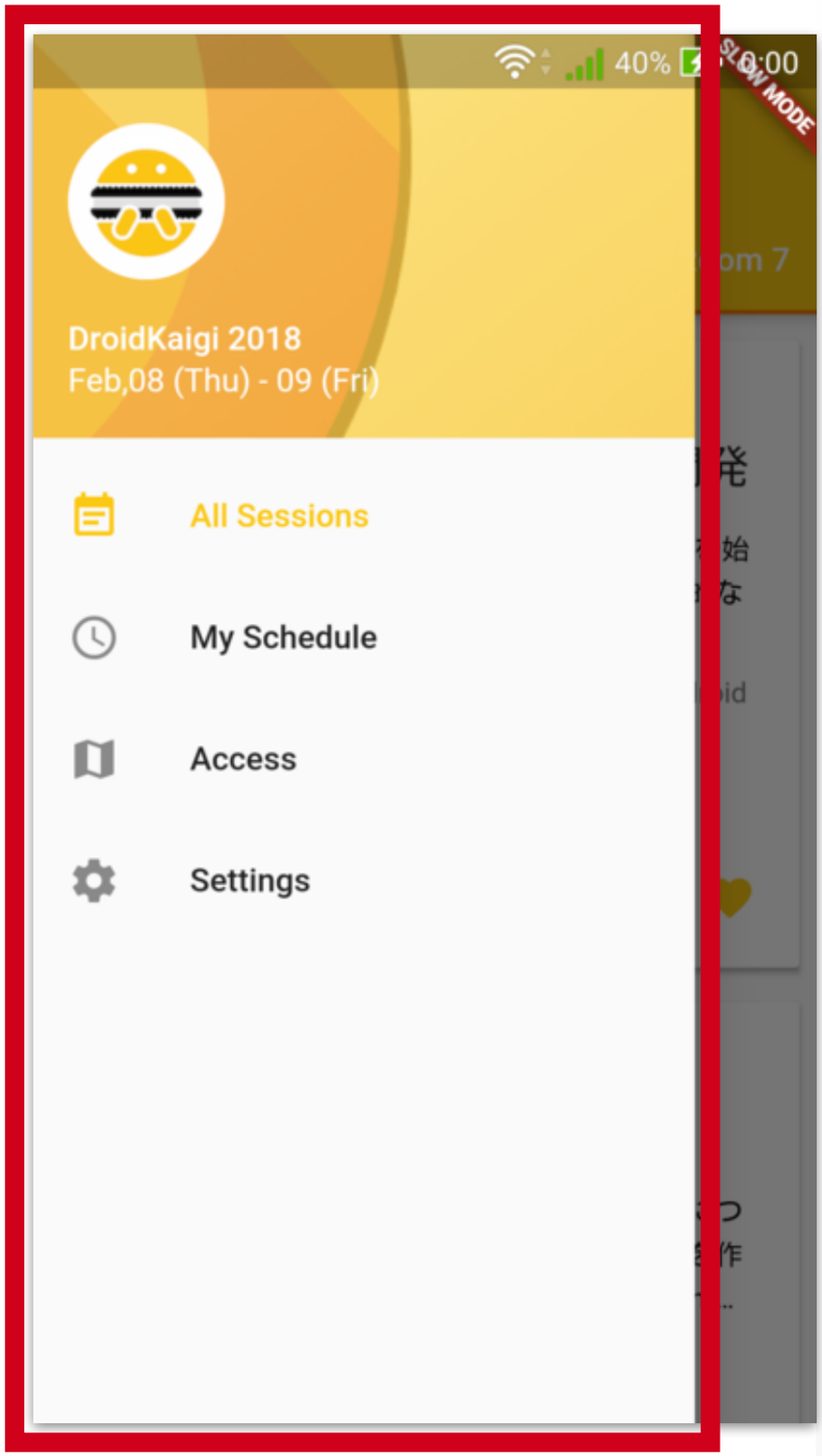
例：Scaffold

AppBar


```
return new Scaffold(
  appBar: new AppBar(
    title: new Text(_title),
    elevation: _page.hasTab ? 0.0 : 4.0,
  ),
  drawer: new MyDrawer(
    items: _pages,
    currentIndex: _currentIndex,
    onTap: (int index) {
      setState(() {
        _page.controller.reverse();
        _currentIndex = index;
        _page.controller.forward();
        _title = _page.title;
      });
    },
  ),
  body: new Center(
    child: _buildTransitionsStack(),
  ),
);
```

例：Scaffold

Drawer




```
return new Scaffold(
  appBar: new AppBar(
    title: new Text(_title),
    elevation: _page.hasTab ? 0.0 : 4.0,
  ),
  drawer: new MyDrawer(
    items: _pages,
    currentIndex: _currentIndex,
    onTap: (int index) {
      setState(() {
        _page.controller.reverse();
        _currentIndex = index;
        _page.controller.forward();
        _title = _page.title;
      });
    },
  ),
  body: new Center(
    child: _buildTransitionsStack(),
  ),
);
```

例：Scaffold


```
return new Scaffold(
  appBar: new AppBar(
    title: new Text(_title),
    elevation: _page.hasTab ? 0.0 : 4.0,
  ),
  drawer: new MyDrawer(
    items: _pages,
    currentIndex: _currentIndex,
    onTap: (int index) {
      setState(() {
        _page.controller.reverse();
        _currentIndex = index;
        _page.controller.forward();
        _title = _page.title;
      });
    },
  ),
  body: new Center(
    child: _buildTransitionsStack(),
  ),
);
```

例：Scaffold

```
class Scaffold extends StatefulWidget {
  /// Creates a visual scaffold for material design widgets.
  const Scaffold({
    Key key,
    this.appBar,
    this.body,
    this.floatingActionButton,
    this.persistentFooterButtons,
    this.drawer,
    this.endDrawer,
    this.bottomNavigationBar,
    this.backgroundColor,
    this.resizeToAvoidBottomPadding: true,
    this.primary: true,
  }) : assert(primary != null), super(key: key);
```

A lots of Widgets!!

Widgets Catalog

Edit Source File Issue

Create beautiful apps faster with Flutter's collection of visual, structural, platform, and interactive widgets.

In addition to browsing widgets by category, you can also see all the widgets in the Flutter widget index.

Basics

Widgets you absolutely need to know before building your first Flutter app.

VISIT

Material Design

Visual, behavioral, and motion-rich widgets implementing Google's Material Design guidelines.

VISIT

Cupertino (iOS-style widgets)

Beautiful and high-fidelity widgets for current iOS design language.

VISIT

Layout

Arrange other widgets columns, rows, grids, and many other layouts.

VISIT

Text

Display and style text.

VISIT

Assets, Images, and Icons

Manage assets, display images, and show icons.

VISIT

Input

Take user input in addition to input

Animation and Motion

Bring animations to your app. Check

Interaction Models

Respond to touch events and route

https://flutter.io/widgets/

Androiderから見たFlutter

- **IntelliJ**で開発できる
- **Dart**はJavaと似てる。標準ライブラリが豊富
- **Hot Reload**で修正が1秒で反映。反映できない時はフルリロードが必要と教えてくれる。フルリロードも10秒くらい（最高）

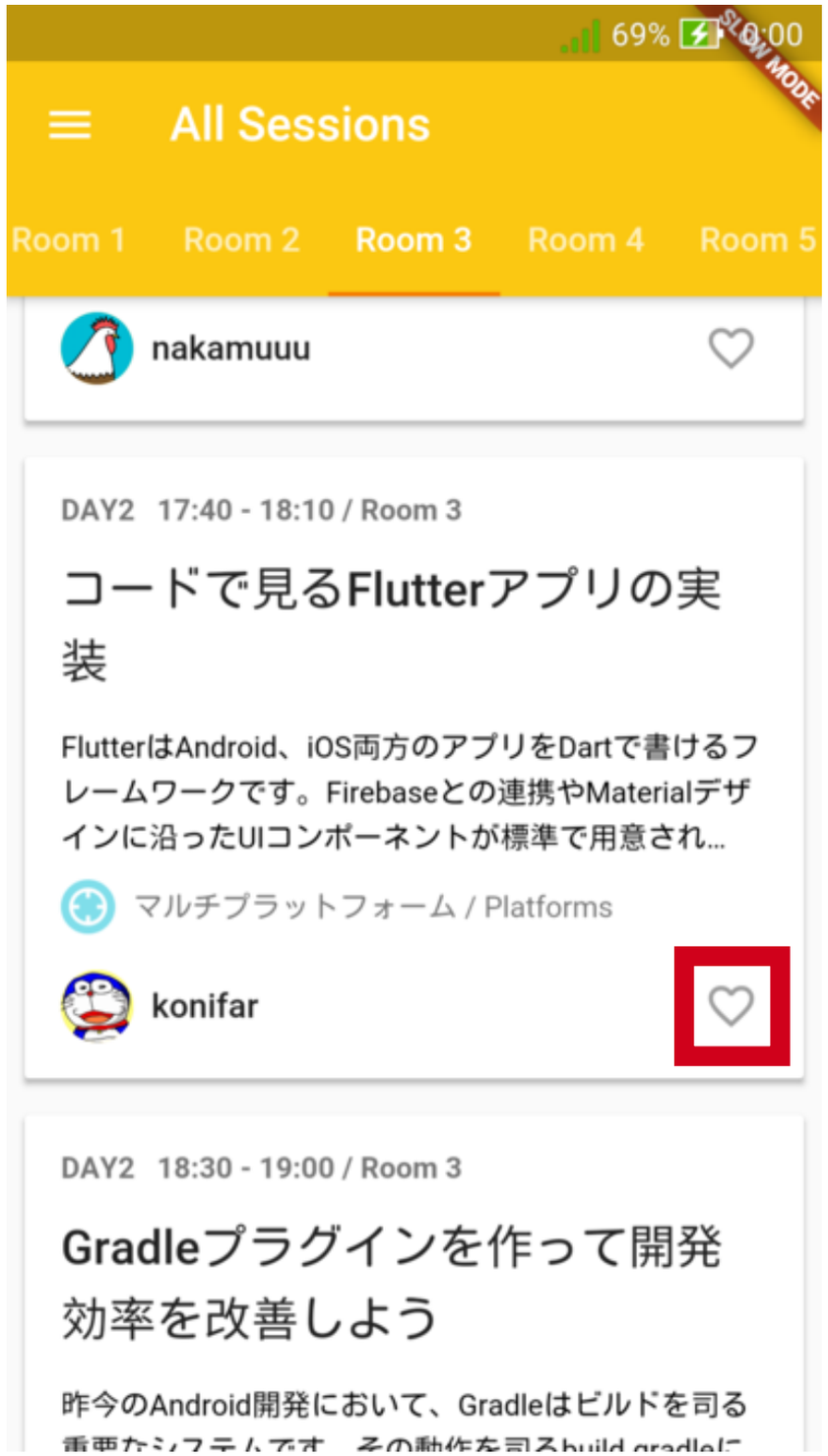
2. コードで見るFlutter

2-1. Widget

Widgetの基本

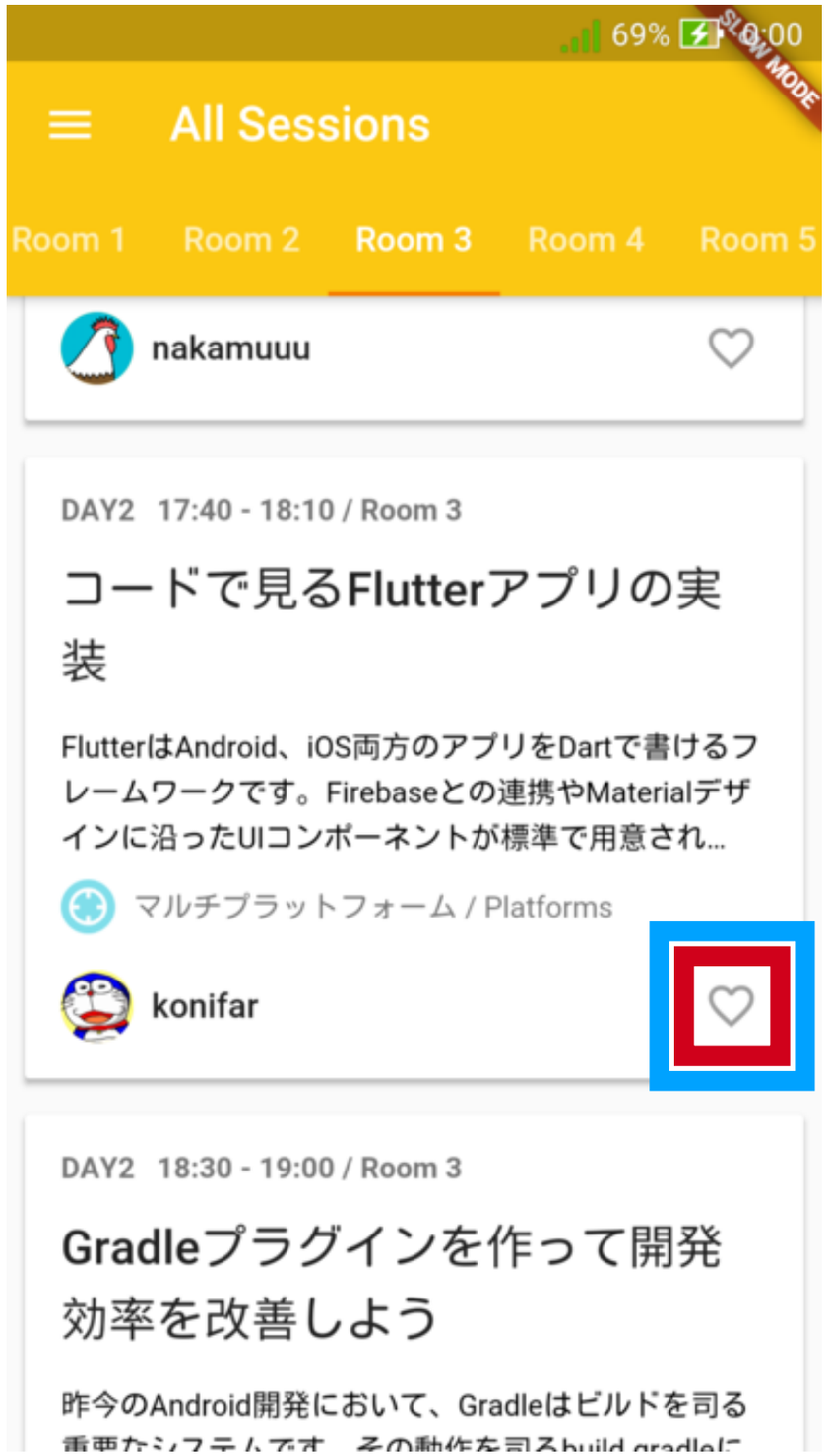
1. すべてはWidgetである
2. statelessとstatefulの二種類がある
3. Widget Treeを組み立てる

すべてはWidgetである



- Icon = Widget

すべてはWidgetである

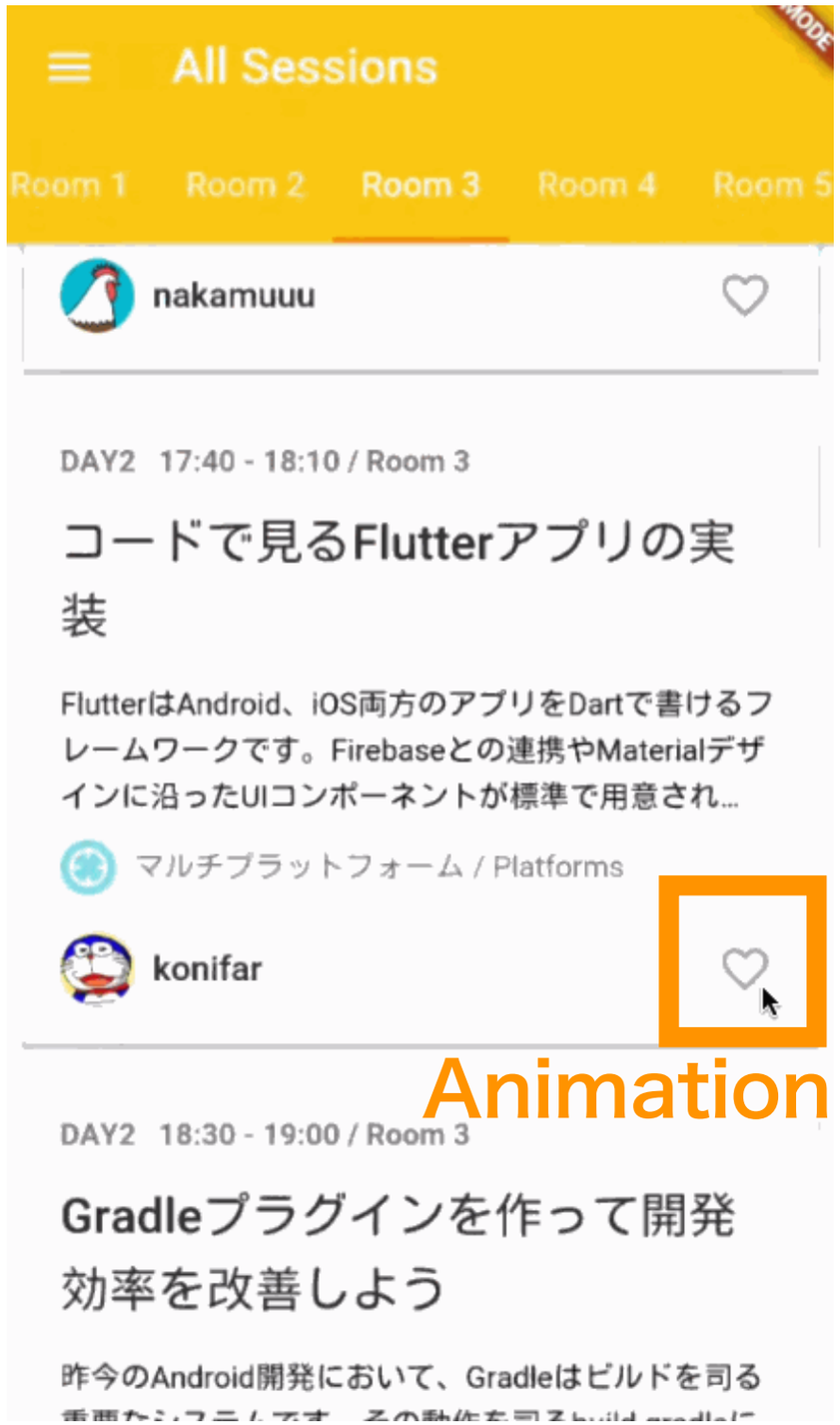


- Icon = Widget
- IconButton = Widget

すべてはWidgetである

- Icon = Widget
- IconButton = Widget
- Positioned = Widget

すべてはWidgetである



- Icon = Widget
- IconButton = Widget
- Positioned = Widget
- ScaleTransition = Widget

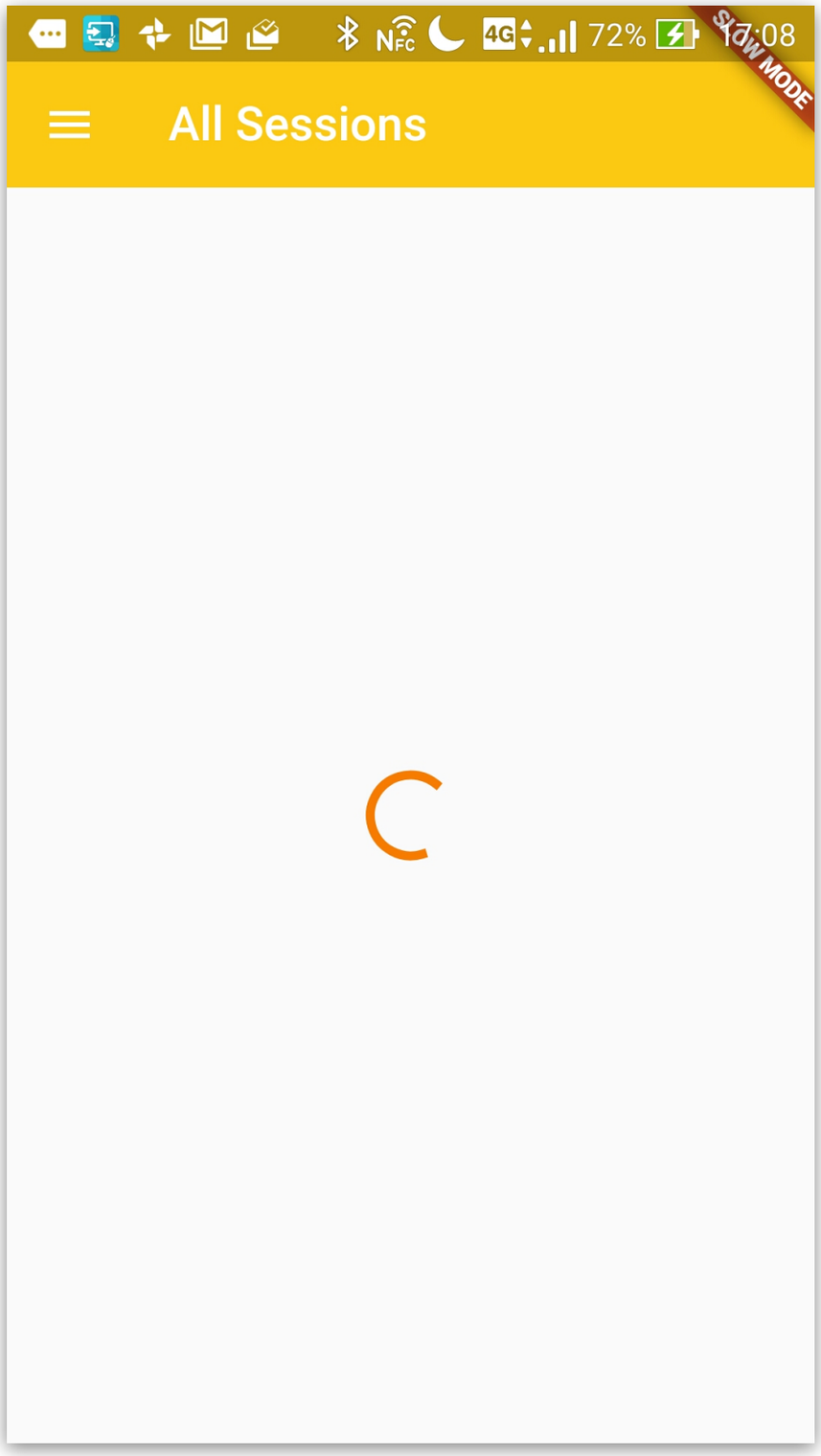
すべてはWidgetである


```
new Positioned(
  bottom: -8.0,
  right: -8.0,
  child: new ScaleTransition(
    scale: new CurvedAnimation(
      parent: animationController,
      curve: Curves.elasticOut,
    ),
    child: new IconButton(
      icon: new Icon(Icons.favorite),
    ),
  ),
);
```

Stateless & Stateful

- 状態を持たないWidgetは**stateless**にする
- 通信結果やユーザーの操作で動的に変更が起きる場合など、stateを持つべきwidgetは**stateful**にする

statefulの例: Loadingの表示



Loadingの表示

```
bool _isLoading = true;
```

Loading state

```
@override
void initState() {
  super.initState();

  new RepositoryFactory().getRoomRepository()
      .findAll()
      .then((r) {
        onDataLoaded(r.values.toList());
      });
}
```

Loadingの表示

```
bool _isLoading = true;

@override
void initState() {
  super.initState();

  new RepositoryFactory().getRoomRepository()
      .findAll()
      .then((r) {
        onDataLoaded(r.values.toList());
      });
}
```

Called only first time

Loadingの表示

```
bool _isLoading = true;

@override
void initState() {
  super.initState();
  new RepositoryFactory().getRoomRepository()
      .findAll()
      .then((r) {
        onDataLoaded(r.values.toList());
      });
}
```

Load the room (tab) list

Loadingの表示

```
bool _isLoading = true;

@override
void initState() {
  super.initState();

  new RepositoryFactory().getRoomRepository()
      .findAll()
      .then((r) {
        onDataLoaded(r.values.toList());
      });
}
```

Called after loading

Loadingの表示

```
void onDataLoaded(List<Room> rooms) {
  setState(() {
    _isLoading = false;
    _rooms = rooms;
    ...
  });
}

@override
Widget build(BuildContext context) {
  if (_isLoading) {
    return new Center(
      child: const CircularProgressIndicator(),
    );
  }
```

Changed state in setState()

Loadingの表示

```
void onDataLoaded(List<Room> rooms) {
  setState(() {
    _isLoading = false;
    _rooms = rooms;
    ...
  });
}
@override
Widget build(BuildContext context) {
  if (_isLoading) {
    return new Center(
      child: const CircularProgressIndicator(),
    );
  }
```

stateが書き換わると

build()が呼ばれてWidgetが書き換わる

ポイント

- どのWidgetがstateを持つべきかを決めるのがいちばん大事

- このTutorialを読むとstateの考え方がわかる

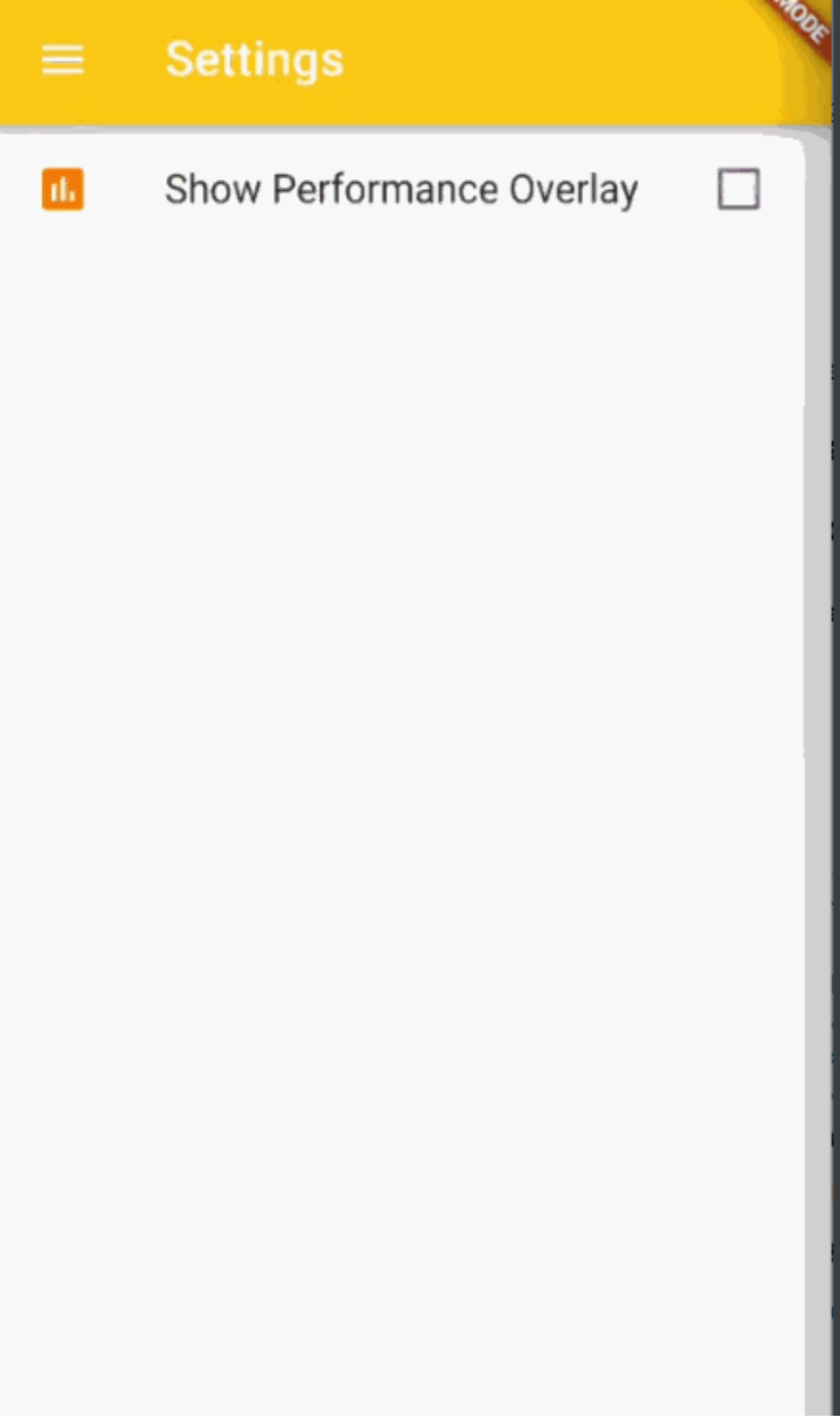
  https://flutter.io/tutorials/interactive

2-2. Widget Tree

Widget Tree

- FlutterではWidgetを入れ子にしてツリーのようにレイアウトを組み立てていく

- IntelliJのFlutter pluginがサポートしてくれる



動画：https://goo.gl/wubWiE

Widget Tree

補完（Option + Enter）を使ってWidget Treeを組んでいき、使い方がわからなければWidgetのクラスのプロパティを見て理解すること

2-3. データの扱い

データの扱い

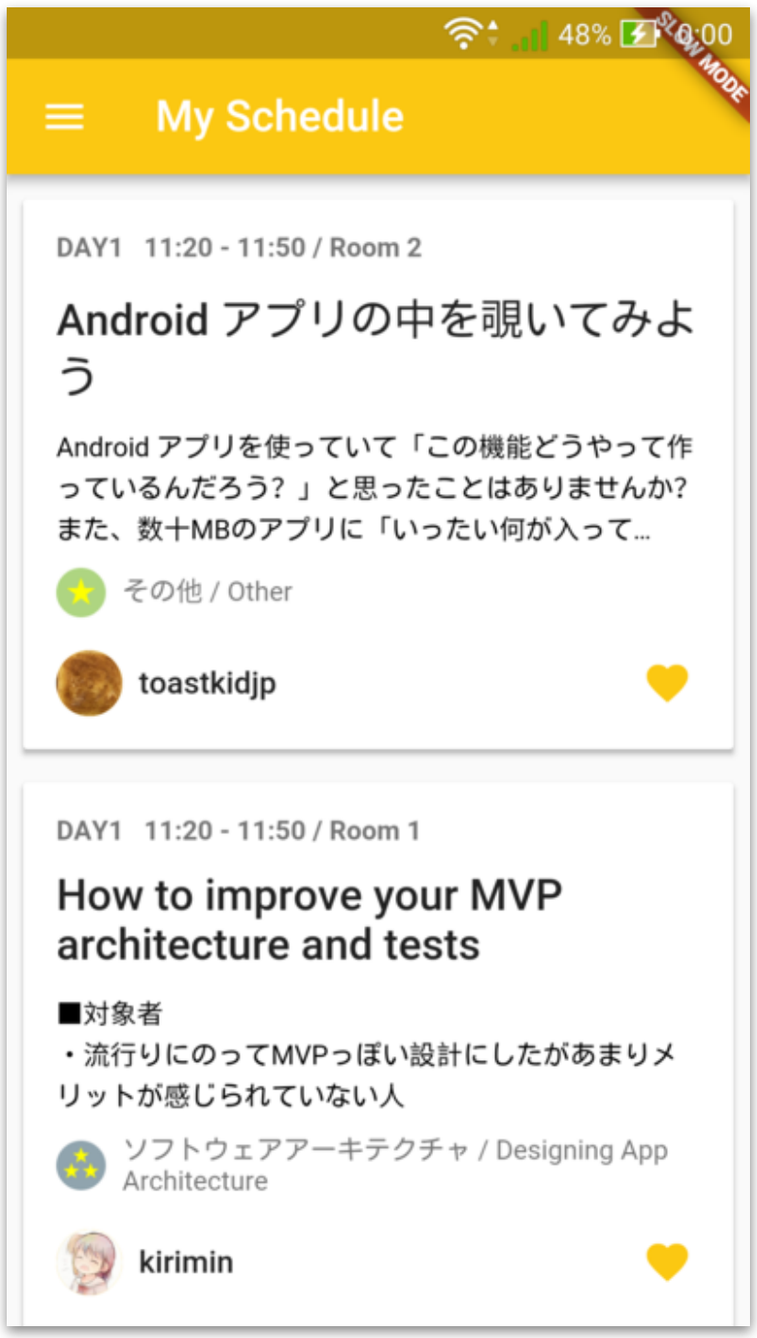
1. ネットワーク経由で取得
2. モデルへの変換
3. ローカルでのキャッシュ

Firebaseをつかう

公式pluginが用意されている

- cloud_firestore
  https://github.com/flutter/plugins/tree/master/packages/cloud_firestore
- firebase_database
  https://github.com/flutter/plugins/tree/master/packages/firebase_database
- firebase_storage
  https://github.com/flutter/plugins/tree/master/packages/firebase_storage

cloud_storeからマイセッション取得



users / $userId / favorites / $sessionId
{ favorite : boolean }

cloud_storeからマイセッション取得

users / $userId / favorites / $sessionId
{ favorite : boolean }

```
Future<Map<String, bool>> findAll(String userId) async {
    ...
    final Map<String, bool> result = new Map();
    final Stream<QuerySnapshot> snapshots =
        _firestore.collection("users/$userId/favorites").snapshots;

    await snapshots.first.then((snapshot) {
      snapshot.documents.forEach((DocumentSnapshot document) {
        String sessionId = document.documentID;
        bool favorite = document.data.values.toList()[0];
        result[sessionId] = favorite;
      });
    });
    ...
}
```

cloud_storeからマイセッション取得

users / $userId / favorites / $sessionId
{ favorite : boolean }

```
Future<Map<String, bool>> findAll(String userId) async {
    ...
    final Map<String, bool> result = new Map();
    final Stream<QuerySnapshot> snapshots =
        _firestore.collection("users/$userId/favorites").snapshots;

    await snapshots.first.then((snapshot) {
      snapshot.documents.forEach((DocumentSnapshot document) {
        String sessionId = document.documentID;
        bool favorite = document.data.values.toList()[0];
        result[sessionId] = favorite;
      });
    });
    ...
}
```

cloud_storeからマイセッション取得

users / $userId / favorites / $sessionId
{ favorite : boolean }

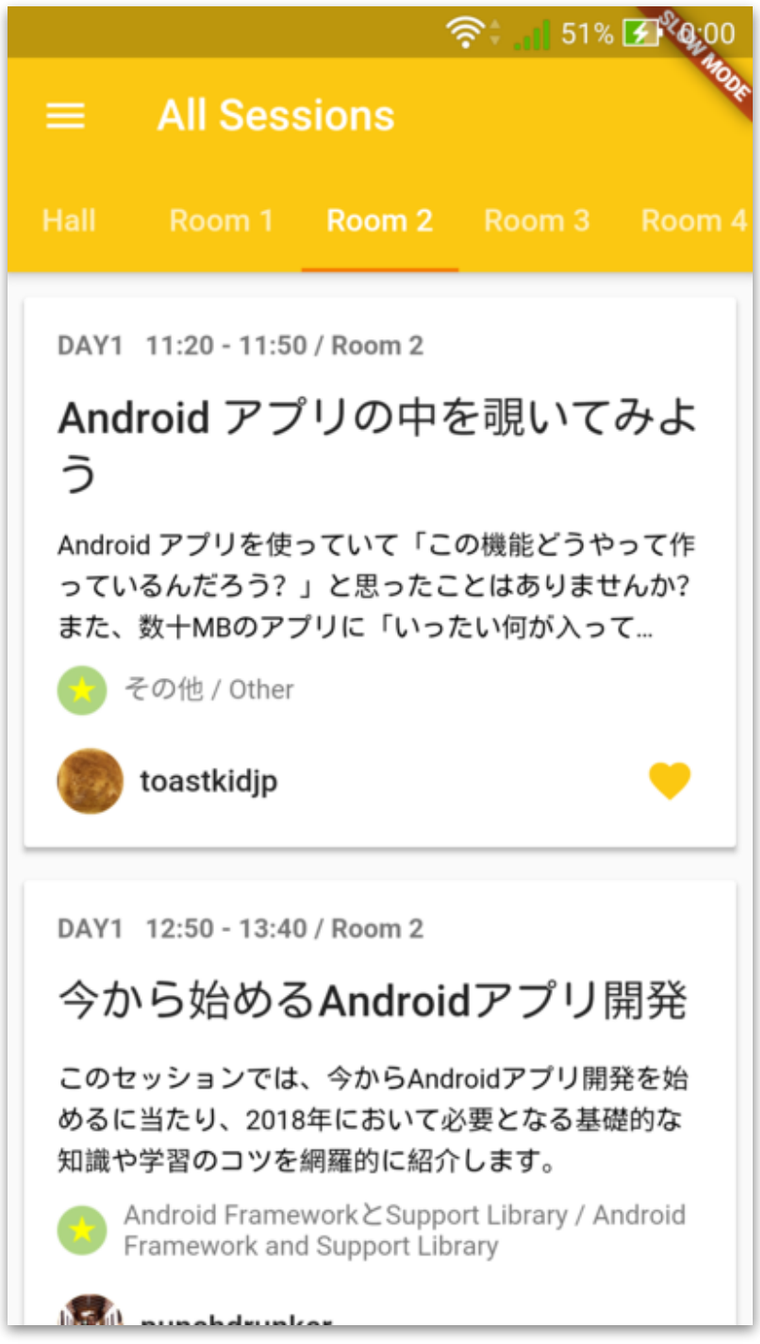
```
Future<Map<String, bool>> findAll(String userId) async {
    ...
    final Map<String, bool> result = new Map();
    final Stream<QuerySnapshot> snapshots =
        _firestore.collection("users/$userId/favorites").snapshots;

    await snapshots.first.then((snapshot) {
      snapshot.documents.forEach((DocumentSnapshot document) {
        String sessionId = document.documentID;
        bool favorite = document.data.values.toList()[0];
        result[sessionId] = favorite;
      });
    });
    ...
}
```

httpライブラリを使う

- Dartのhttp package + async, await

- dart:convert でパース

セッション一覧の取得



https://droidkaigi.jp/2018/sessionize/all.json


```
{
 "sessions": [
  {
   "id": "16969",
   "title": "Kotlinアンチパターン",
   "description": " KotlinはJavaよりもシンプルで安全なコードを書くことができます。また、
すが、使いどころ、特にAndroidアプリ開発における使いどころについては、まだそれぞれ模索し
開発した経験の中で分かった、Kotlinの各種文法の適切な使いどころや、バグを生みやすいコー
デフォルト実装とabstractクラスの使い分け、引数なしの関数とcustom getterの使い分け、定
関数の使いどころ、レシーバー付き関数型の使いどころなど、Kotlinの基本文法を紹介しつつ、A
注意点などについて解説してみたいと思います。",
   "startsAt": "2018-02-08T10:20:00",
   "endsAt": "2018-02-08T10:50:00",
   "isServiceSession": false,
   "isPlenumSession": false,
   "speakers": [
    "3fe8f4be-5700-44f5-ba97-1a4b4f831ec8"
   ],
   "categoryItems": [
    3480,
    3483,
    3488,
    3541
   ],
   "questionAnswers": [],
   "roomId": 513
  },
```

セッション一覧の取得

```
static const _BASE_URL = 'https://droidkaigi.jp/2018/sessionize';

_requestAll() async {
  var response = await http.read("$_BASE_URL/all.json");
  var json = JSON.decode(response);
  ...
```

セッション一覧の取得

```
static const _BASE_URL = 'https://droidkaigi.jp/2018/sessionize';

_requestAll() async {
  var response = await http.read("$_BASE_URL/all.json");
  var json = JSON.decode(response);
  ...
```

モデルへのマッピング

- convertライブラリを使って手動でやるか
- built_valueを使ってdeserializeするか

  https://github.com/google/built_value.dart

Preferenceへの値保存

- 公式のshared_preferences pluginがある

  https://github.com/flutter/plugins/tree/master/packages/shared_preferences

- Android/iOSともに動く

前回開いたタブ位置の保存

```dart
const String PREF_KEY_TAB_INDEX = "pref_key_tab_index";

_saveTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  print("saveTabIndex: ${_controller?.index}");
  prefs.setInt(PREF_KEY_TAB_INDEX, _controller?.index);
}

Future<int> _restoreTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int tabIndex = prefs.getInt(PREF_KEY_TAB_INDEX);
  return tabIndex != null ? tabIndex : _controller?.index;
}
```

前回開いたタブ位置の保存

```dart
const String PREF_KEY_TAB_INDEX = "pref_key_tab_index";

_saveTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  print("saveTabIndex: ${_controller?.index}");
  prefs.setInt(PREF_KEY_TAB_INDEX, _controller?.index);
}

Future<int> _restoreTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int tabIndex = prefs.getInt(PREF_KEY_TAB_INDEX);
  return tabIndex != null ? tabIndex : _controller?.index;
}
```

前回開いたタブ位置の保存

```
const String PREF_KEY_TAB_INDEX = "pref_key_tab_index";

_saveTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  print("saveTabIndex: ${_controller?.index}");
  prefs.setInt(PREF_KEY_TAB_INDEX, _controller?.index);
}

Future<int> _restoreTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int tabIndex = prefs.getInt(PREF_KEY_TAB_INDEX);
  return tabIndex != null ? tabIndex : _controller?.index;
}
```

前回開いたタブ位置の保存

```
const String PREF_KEY_TAB_INDEX = "pref_key_tab_index";

_saveTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  print("saveTabIndex: ${_controller?.index}");
  prefs.setInt(PREF_KEY_TAB_INDEX, _controller?.index);
}

Future<int> _restoreTabIndex() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int tabIndex = prefs.getInt(PREF_KEY_TAB_INDEX);
  return tabIndex != null ? tabIndex : _controller?.index;
}
```

SQLite

- sqfliteでSQLiteを操作する（Android / iOS）
  https://github.com/tekartik/sqflite
- Raw SQLの実行に加えて、Insert、Delete、Updateなどの操作のHelperが用意されている

3. 一問一答

Q1. CIまわせる？

A. Yes

ただし、Flutterは x86_64-linux-gnu/libstdc++.so.6 version GLIBCXX_3.4.18 が必要なので、同バージョンの環境でCIを回す必要がある。

Travisの例

```yaml
os:
  - linux
sudo: false
addons:
  apt:
    sources:
      - ubuntu-toolchain-r-test
    packages:
      - libstdc++6
      - fonts-droid
before_script:
  - git clone https://github.com/flutter/flutter.git -b alpha --depth 1
  - ./flutter/bin/flutter doctor
script:
  - ./flutter/bin/flutter test
cache:
  directories:
    - $HOME/.pub-cache
```

Travisの例

```yaml
os:
  - linux
sudo: false
addons:
  apt:
    sources:
      - ubuntu-toolchain-r-test
    packages:
      - libstdc++6
      - fonts-droid
before_script:
  - git clone https://github.com/flutter/flutter.git -b alpha --depth 1
  - ./flutter/bin/flutter doctor
script:
  - ./flutter/bin/flutter test
cache:
  directories:
    - $HOME/.pub-cache
```

Travisの例

```
os:
  - linux
sudo: false
addons:
  apt:
    sources:
      - ubuntu-toolchain-r-test
    packages:
      - libstdc++6
      - fonts-droid
before_script:
  - git clone https://github.com/flutter/flutter.git -b alpha --depth 1
  - ./flutter/bin/flutter doctor
script:
  - ./flutter/bin/flutter test
cache:
  directories:
    - $HOME/.pub-cache
```

Setup Flutter

Travisの例

```
os:
  - linux
sudo: false
addons:
  apt:
    sources:
      - ubuntu-toolchain-r-test
    packages:
      - libstdc++6
      - fonts-droid
before_script:
  - git clone https://github.com/flutter/flutter.git -b alpha --depth 1
  - ./flutter/bin/flutter doctor
script:
  - ./flutter/bin/flutter test
cache:
  directories:
    - $HOME/.pub-cache
```

Run test

Q2. Analyticsどうするの？

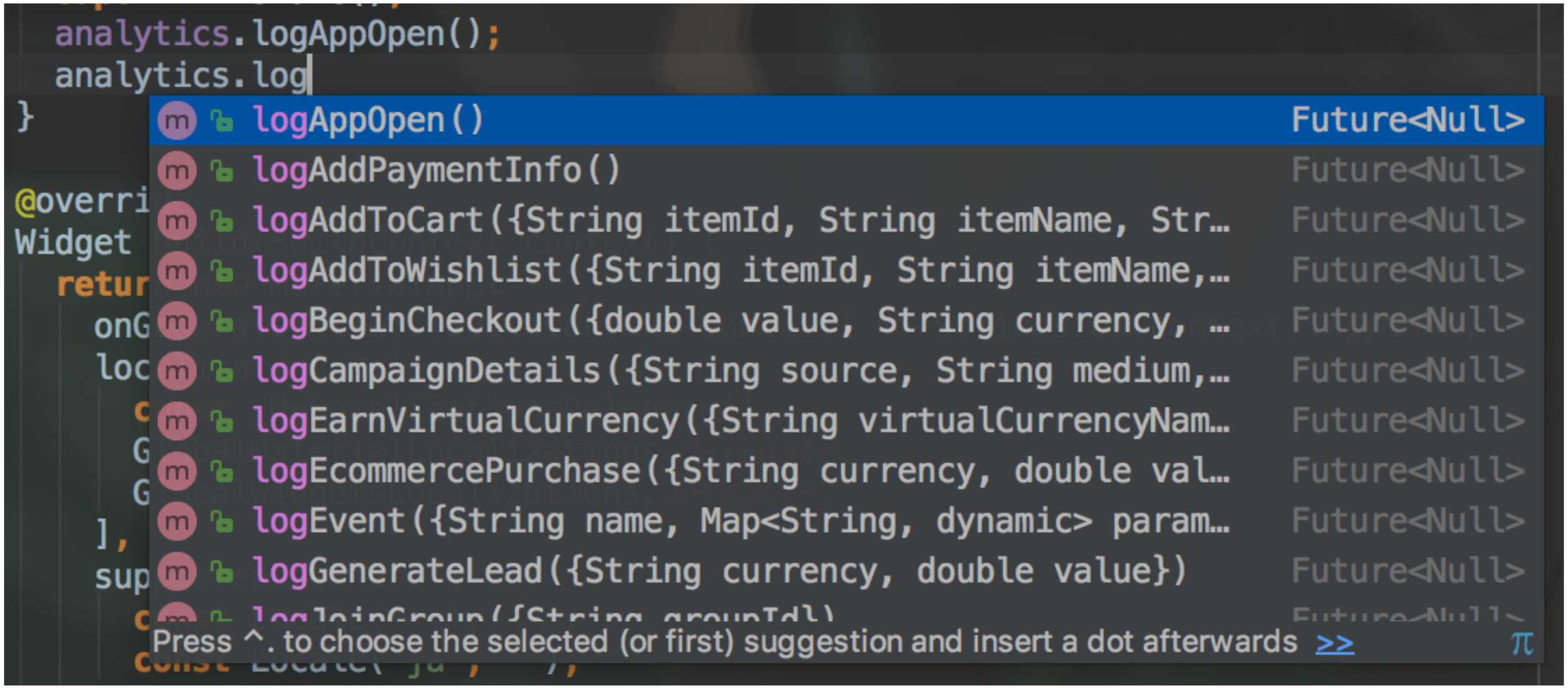
A. FirebaseAnalytics pluginをつかう

https://github.com/flutter/plugins/tree/master/packages/firebase_analytics

ページ遷移のトラッキング

```
FirebaseAnalytics analytics = new FirebaseAnalytics();

MaterialApp(
  home: new MyAppHome(),
  navigatorObservers: [
    new FirebaseAnalyticsObserver(analytics: analytics),
  ],
);
```

ログの送信


analytics.logAppOpen();
analytics.log
}
@overri
Widget
retur
onG
loc
],
sup
logAppOpen() Future<Null>
logAddPaymentInfo() Future<Null>
logAddToCart({String itemId, String itemName, Str… Future<Null>
logAddToWishlist({String itemId, String itemName,… Future<Null>
logBeginCheckout({double value, String currency, … Future<Null>
logCampaignDetails({String source, String medium,… Future<Null>
logEarnVirtualCurrency({String virtualCurrencyNam… Future<Null>
logEcommercePurchase({String currency, double val… Future<Null>
logEvent({String name, Map<String, dynamic> param… Future<Null>
logGenerateLead({String currency, double value}) Future<Null>
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards >>

Q3. Push Notificationはどうする？

A. FirebaseMessaging pluginをつかう

https://github.com/flutter/plugins/tree/master/packages/firebase_messaging

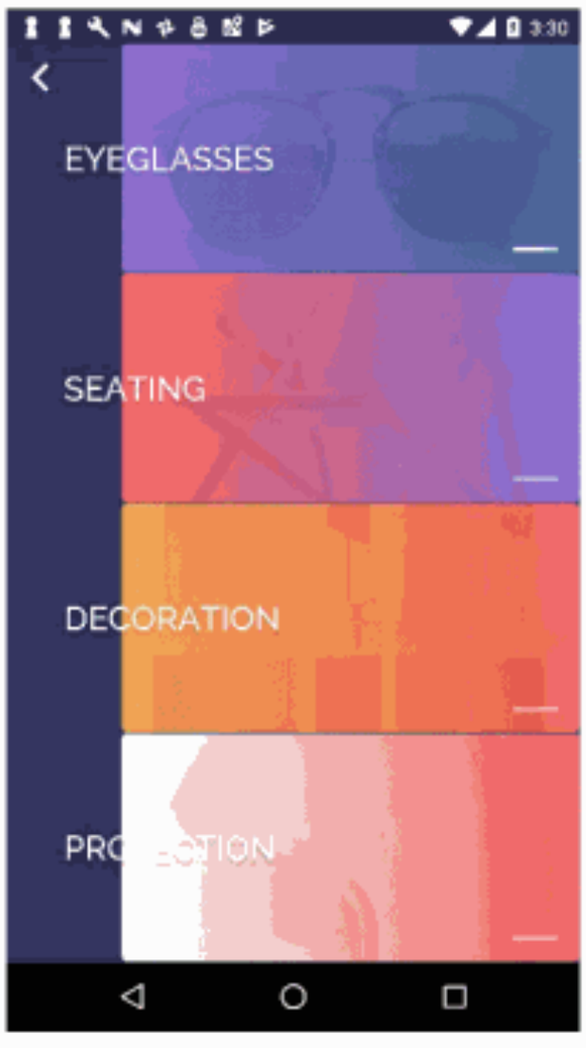
Q4. Animationはサポートされてる？

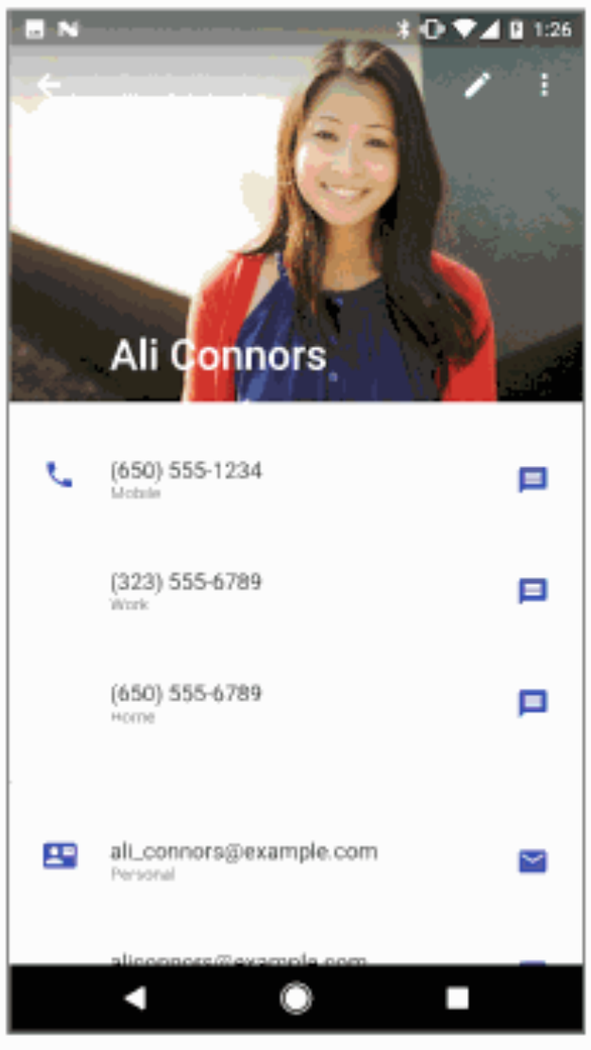
A. Yes

Animation、TransitionもWidgetで表現するので、やり方はAndroidとだいぶ違う

https://flutter.io/animations/

Animation Examples





動画：https://goo.gl/npYoj1

Q5. ユニットテストどう書く？

A. mockito.dart、test/test.dart をつかう

https://flutter.io/testing/

Jsonからモデルにパースするテスト

```
void main() {
  group("fromJson", () {
    test('works well.', () {
      // given
      var json = JSON.decode("""
{
  "linkType": "Twitter",
  "url": "https://twitter.com/konifar"
}
      """);

      // when
      Link link = Link.fromJson(json);

      // then
      expect(link.type, LinkType.twitter);
      expect(link.url, 'https://twitter.com/konifar');
    });
  });
}
```

Jsonからモデルにパースするテスト

```
void main() {
  group("fromJson", () {
    test('works well.', () {
      // given
      var json = JSON.decode("""
{
  "linkType": "Twitter",
  "url": "https://twitter.com/konifar"
}
      """);

      // when
      Link link = Link.fromJson(json);

      // then
      expect(link.type, LinkType.twitter);
      expect(link.url, 'https://twitter.com/konifar');
    });
  });
}
```

Jsonからモデルにパースするテスト

```
void main() {
  group("fromJson", () {
    test('works well.', () {
      // given
      var json = JSON.decode("""
{
  "linkType": "Twitter",
  "url": "https://twitter.com/konifar"
}
      """);

      // when
      Link link = Link.fromJson(json);

      // then
      expect(link.type, LinkType.twitter);
      expect(link.url, 'https://twitter.com/konifar');
    });
  });
}
```

Q6. 多言語化どうやるの？

A. dart:intlをつかう

https://flutter.io/tutorials/internationalization/

ただし、strings.xmlと比べてわりと面倒

Widgetのテストがコケるので、tester.pump()を呼んでおくワークアラウンドが必要

https://github.com/flutter/flutter/issues/1865

Q7. クラッシュログの収集方法は？

A. Firebase Crash Reporting または Sentryをつかう

https://github.com/flutter/flutter/issues/614

https://firebase.google.com/docs/crash

https://pub.dartlang.org/packages/sentry

Q8. ライブラリを探すときは？

A. 公式プラグイン or dart libraryを検索しよう

https://github.com/flutter/plugins

https://pub.dartlang.org/flutter/packages

flutter / plugins

Watch 68 | Unstar 245 | Fork 144

Code | Pull requests 6 | Insights

Branch: master ▾ plugins / packages /

Create new file | Upload files | Find file | History

lukef and mravn-google Add path getter to storage ref object (#379) — Latest commit 1c6bc02 8 hours ago

..		
android_alarm_manager	Remove all the examples' podfiles (#340)	9 days ago
android_intent	Remove all the examples' podfiles (#340)	9 days ago
battery	Remove all the examples' podfiles (#340)	9 days ago
camera	Remove all the examples' podfiles (#340)	9 days ago
cloud_firestore	Fix cloud firestore typos	2 days ago
connectivity	Remove all the examples' podfiles (#340)	9 days ago
device_info	Remove all the examples' podfiles (#340)	9 days ago
firebase_admob	Remove all the examples' podfiles (#340)	9 days ago
firebase_analytics	Clarify Google Analytics (#377)	8 days ago
firebase_auth	Added password reset function to Firebase Auth object (#378)	3 days ago
firebase_core	Remove all the examples' podfiles (#340)	9 days ago
firebase_database	Remove all the examples' podfiles (#340)	9 days ago
firebase_messaging	Remove all the examples' podfiles (#340)	9 days ago
firebase_storage	Add path getter to storage ref object (#379)	8 hours ago
google_sign_in	Remove all the examples' podfiles (#340)	9 days ago
image_picker	Small README addition	7 days ago
local_auth	Remove all the examples' podfiles (#340)	9 days ago
package_info	Remove all the examples' podfiles (#340)	9 days ago
path_provider	Remove all the examples' podfiles (#340)	9 days ago
quick_actions	Remove all the examples' podfiles (#340)	9 days ago
sensors	Remove all the examples' podfiles (#340)	9 days ago
share	Remove all the examples' podfiles (#340)	9 days ago
shared_preferences	add a dynamic get method to shared_preferences (#375)	3 days ago

Search Flutter packages

FLUTTER WEB ALL

Top Flutter packages

Sorted by: listing relevance

firebase_auth

Flutter plugin for Firebase Auth, enabling Android and iOS authentication using passwords, and identity providers like Google, Facebook, and Twitter.

v 0.4.4 • Updated: Feb 5, 2018 FLUTTER

100

url_launcher

Flutter plugin for launching a URL on Android and iOS. Supports web, phone, SMS, and email schemes.

v 2.0.1 • Updated: Jan 12, 2018 FLUTTER

100

path_provider

Flutter plugin for getting commonly used locations on the Android & iOS file systems, such as the temp and app data directories.

v 0.3.1 • Updated: Jan 12, 2018 FLUTTER

100

firebase_database

Flutter plugin for Firebase Database, a cloud-hosted NoSQL database with realtime data syncing across Android and iOS clients, and offline access.

v 0.3.4 • Updated: Jan 17, 2018 FLUTTER

99

google_sign_in

Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account on Android and iOS.

100

Lots of Libraries!

まとめ

Flutterのポイント

- どんなWidgetがあるか知るのが大事
- 爆速開発のためにIntelliJを使いこなそう
- 公式ドキュメントとサンプルが豊富なので参考にしよう

“業務”でつかえるのか？

つかえそう

結局プロダクションに
突っ込んでみないとわからない

要するに ”覚悟” 次第

Kyash Inc. にて、虎視眈々と

ぶっこみの機会を伺っています

Thanks!