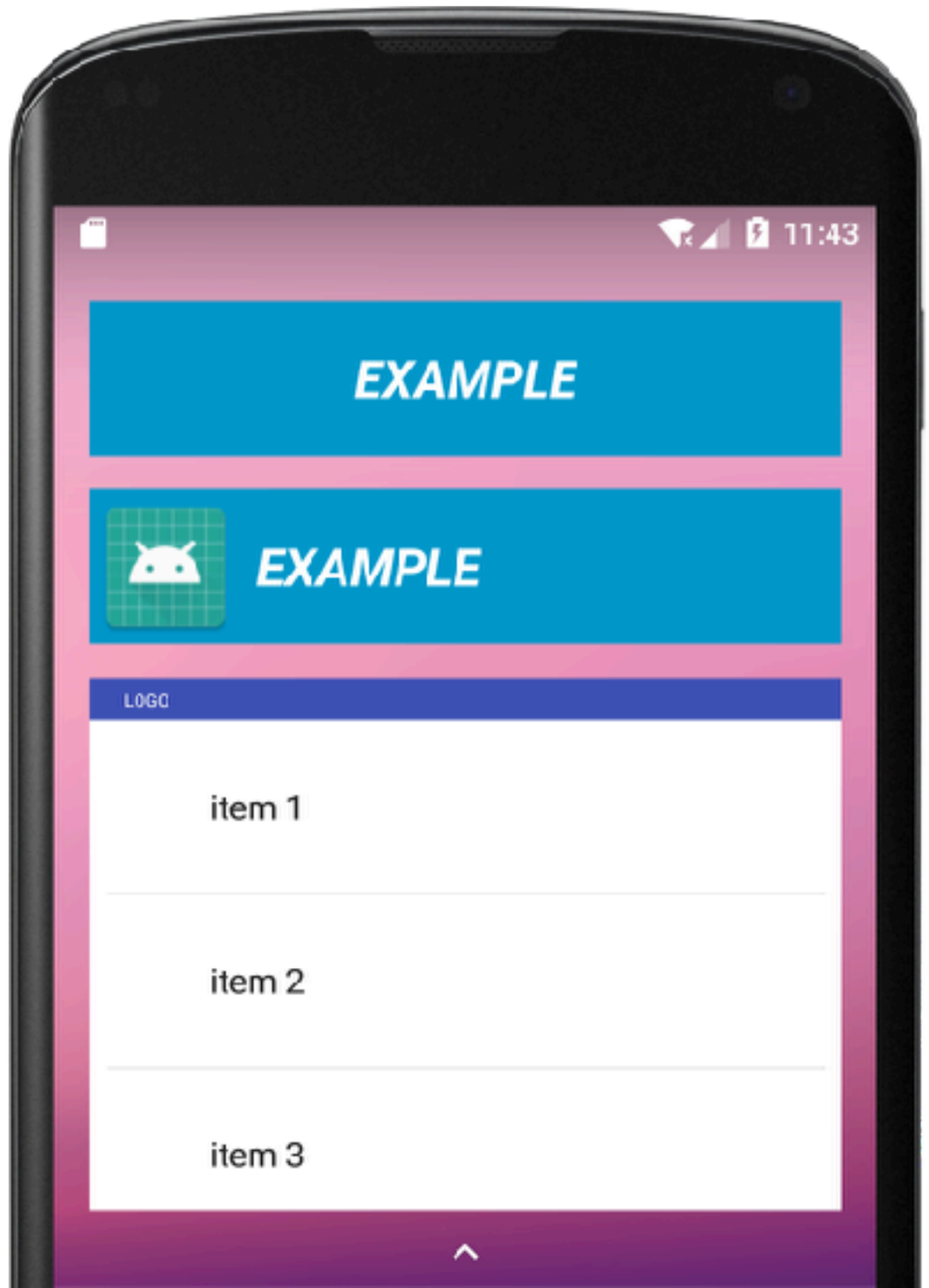
Widget開発再訪

re-introduction to Widget Development
Room 2 — 2018/02/08 14:00-14:30 @ymnd

Sample App

https://github.com/ymnder/WidgetSample



Today's menu

1. About Widget:ウィジェットとは？
2. Widget Design:今日どんなウィジェットがあるか
3. Let's make Widget:つくってワクワク☆
4. New API: お待ちかね新API

About Widget

What Widget

- Widgetはミニアプリ
  - メインのアプリにバンドルして提供される
- アプリの機能やコンテンツにホーム画面から触れる
  - アプリを開かなくてもアクセスできる

Why Widget: for User

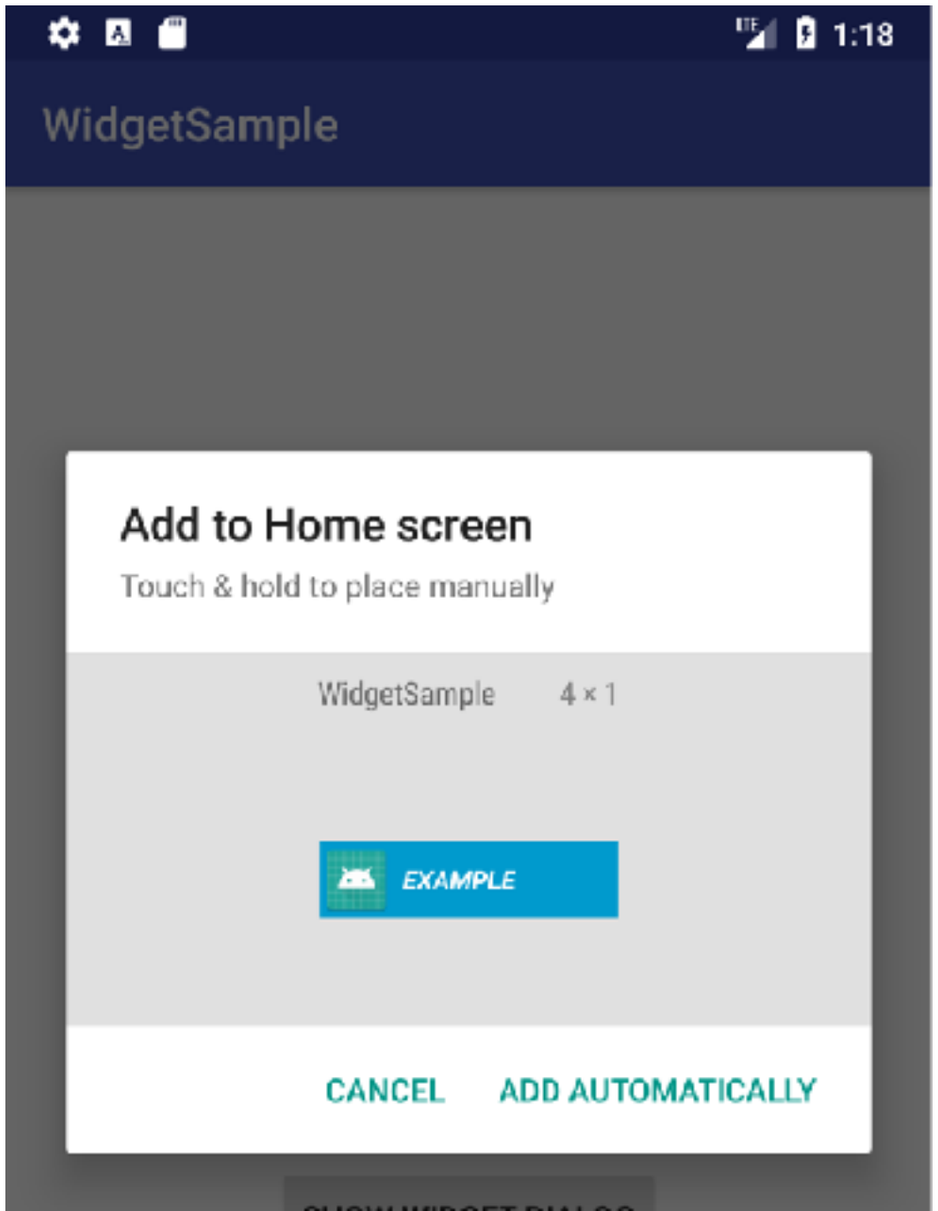
- Widgetはアプリのコア機能を使えるようにしたもの
- アプリを開かなくてもアプリにリーチしてくれる
- ホーム画面でコンテンツを目にする機会が増える

Why Widget now?

- Widgetはアプリの外でコンテンツを触れてもらえる
- 開発に簡単に取り掛かれる
- AndroidOから新しいAPIが追加された

What is New Api

- Widget Install Dialog
- Widgetのインストールフローが改善された



Widget picker

- Storeインストール後に、自分で追加する必要がある
- ホーム画面をロングタップする必要がある
- キャリアのランチャーごとに微妙に導線が異なる
- 他のアプリと同じところにすべて格納されているため探す手間がかかる

Widget picker //TODO:video

Widget Install Dialog

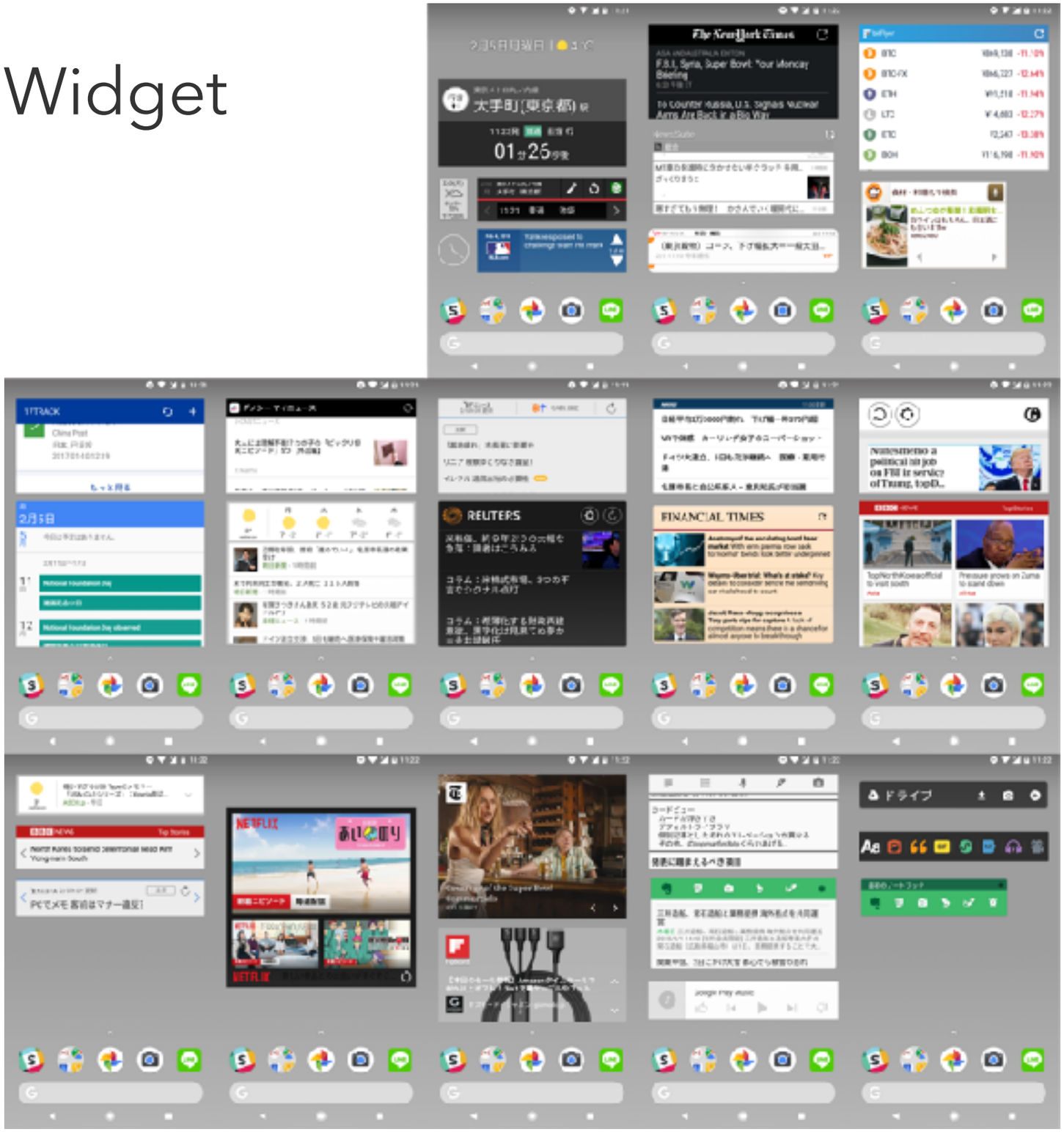
- アプリの中から直接Widgetをはりつけることができる
- 登録フローが劇的に分かりやすくなった！！
- 実際に見てみましょう

Widget Install Dialog //TODO:video

Widget Design

どんなWidgetがあるか

- いろんなWidget



Widget Category

- Information widgets
- Collection widgets
- Control widgets
- Hybrid widgets

Information widgets

- 時計・天気などシンプルな情報を出す
- タップしたら詳細な情報をAPPで表示する

Collection widgets

- 記事などの同じタイプの複数のアイテムをまとめる
- コレクションを一覧する
- タップして詳細を確認する

Control widgets

- アプリをコントロールする
- 再生プレーヤーのボタンや、機能を集約したランチャーのようなもの

Hybrid widgets

- 上の要素を組み合わせたもの
- 例：音楽プレーヤーのランチャー＋ジャケット表示

Widget Design: News Widget Case

Layout

① 一行型

- 一行でニュースをコンパクトに表示
- 次々にニュースを流す

② リスト型

- 記事をリストで表示する、GridよりListが多い

③ ウォールペーパー型

- 写真を見せたい場合に壁紙のように表示する

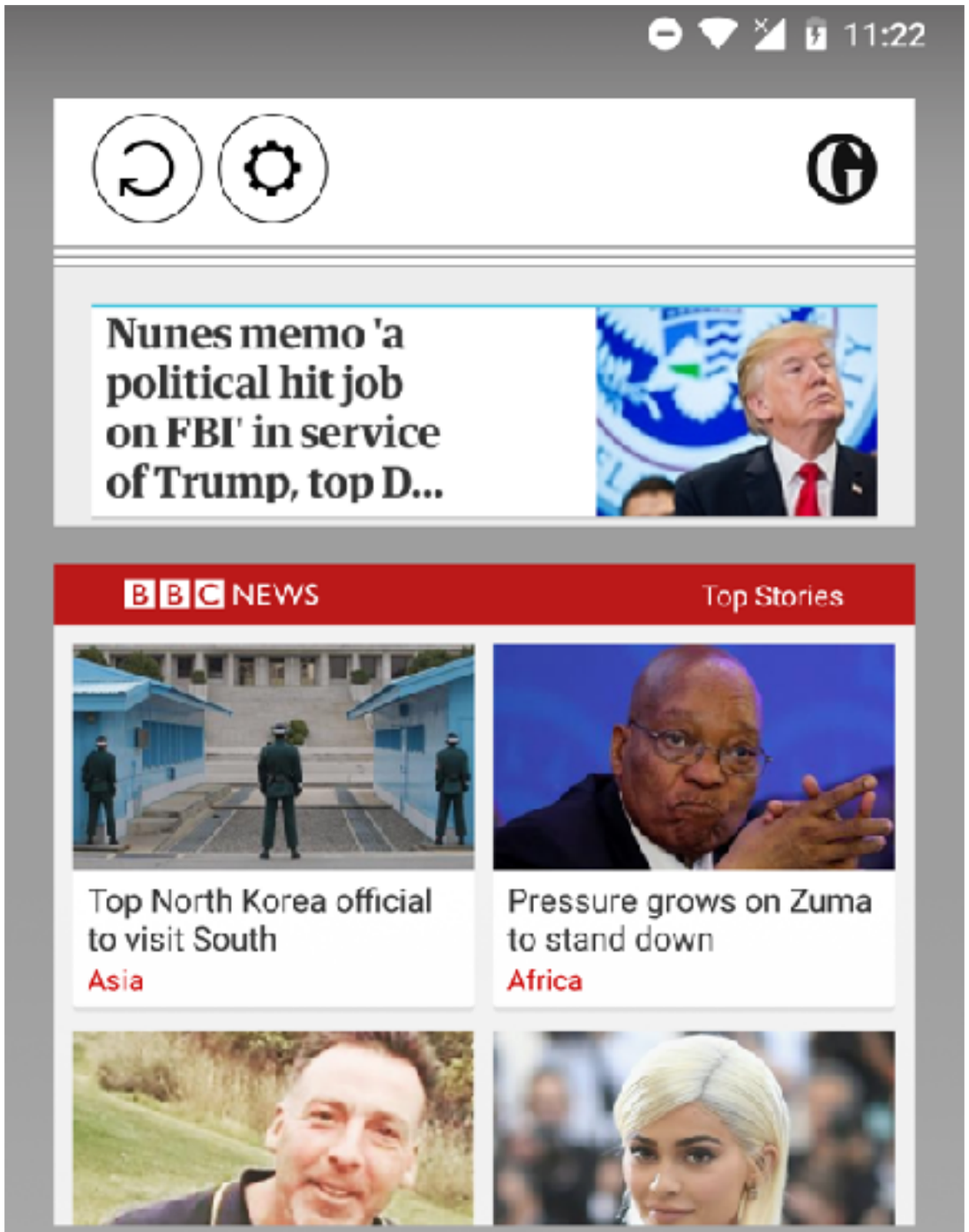
Layout: 一行型


11:22
極小すぎるUSB Type-Cメモリー「USM-CA1シリーズ」：Xperia周辺...
ASCII.jp - 昨日
3°
BBC NEWS
Top Stories
North Korea to send ceremonial head Kim Yong-nam South
Yahoo!ニュース 2/5 9:09 更新
主要
PCでメモ 客前はマナー違反?

Layout: リスト型


11:21
Yahoo!ニュース
2/5 9:09 更新
地域を設定
主要
「基地痘れ」市長選に影響か
リニア 検察ゆとりなさ露呈?
インフル 遠隔出社の必要性
NEW
REUTERS
米株価、約９年ぶりの大幅な急落：識者はこうみる
コラム：米株式市場、3つの不吉なシグナル点灯
コラム：希薄化する財政再建意欲、黒字化は見果てぬ夢か＝永井靖敏氏
11:21
NIKKEI
11:00更新
日経平均2万3000円割れ　下げ幅一時570円超
VRで体感　カーリング女子のスーパーショット
ドイツ大連立、5日も交渉継続へ　医療・雇用で溝
名護市長に自公系新人・渡具知氏が初当選
FINANCIAL TIMES
Anatomy of the escalating bond bear market With term premia now back to 'normal' bonds look better underpinned
Waymo-Uber trial: What's at stake? Key details to consider before the self-driving car rivals head to court
Jacob Rees-Mogg recognises a Tory party ripe for capture A lack of competition means there is a chance for almost anyone to break through
11:22
Nunes memo 'a political hit job on FBI' in service of Trump, top D...
BBC NEWS
Top Stories
Top North Korea official to visit South
Asia
Pressure grows on Zuma to stand down
Africa

Layout: ウォールペーパー型



Configuration

- 色(白地・黒地)
- 更新頻度 (30min, 1h, more)
- ニュースの内容 (top, ranking...)

Widgetの構成要素

① 企業ロゴ

② 設定ボタン

③ 更新ボタン

④ 最終更新日時

⑤ 記事本体 -> tapでアプリを開く

Let's make Widget

Widget Classes

- AppWidgetService
  WidgetのサービスWidgetIdの発行やランチャーへのリクエストを行う
- AppWidgetHost
  ホーム画面のようなWidgetをUIに埋め込みたいAppWidgetサービスとのやりとりを提供する
- AppWidgetHostView
  AppWidgetビューを表示するための接着剤を提供する

---ここから上は直接触らない---

- AppWidgetManager
  Widgetの状態を更新したり、Widgetの情報を取得する管理クラス
- AppWidgetProvider
  providerを実装するのに使う便利なクラス、BroadcastReceiverをwrapしている
- AppWidgetProviderInfo
  providerのメタデータ。appwidget-providerのxmlタグのフィールドに対応している

Widget Classes: Simplify

- AppWidgetProviderInfo
  - Widgetの基本情報をxmlに記述する
- AppWidgetProvider
  - Widgetの更新をハンドルする
- xml/layout
  - Widgetのレイアウト、通常のレイアウトと概ね同じ

How to make it

① manifestにreceiverを追加

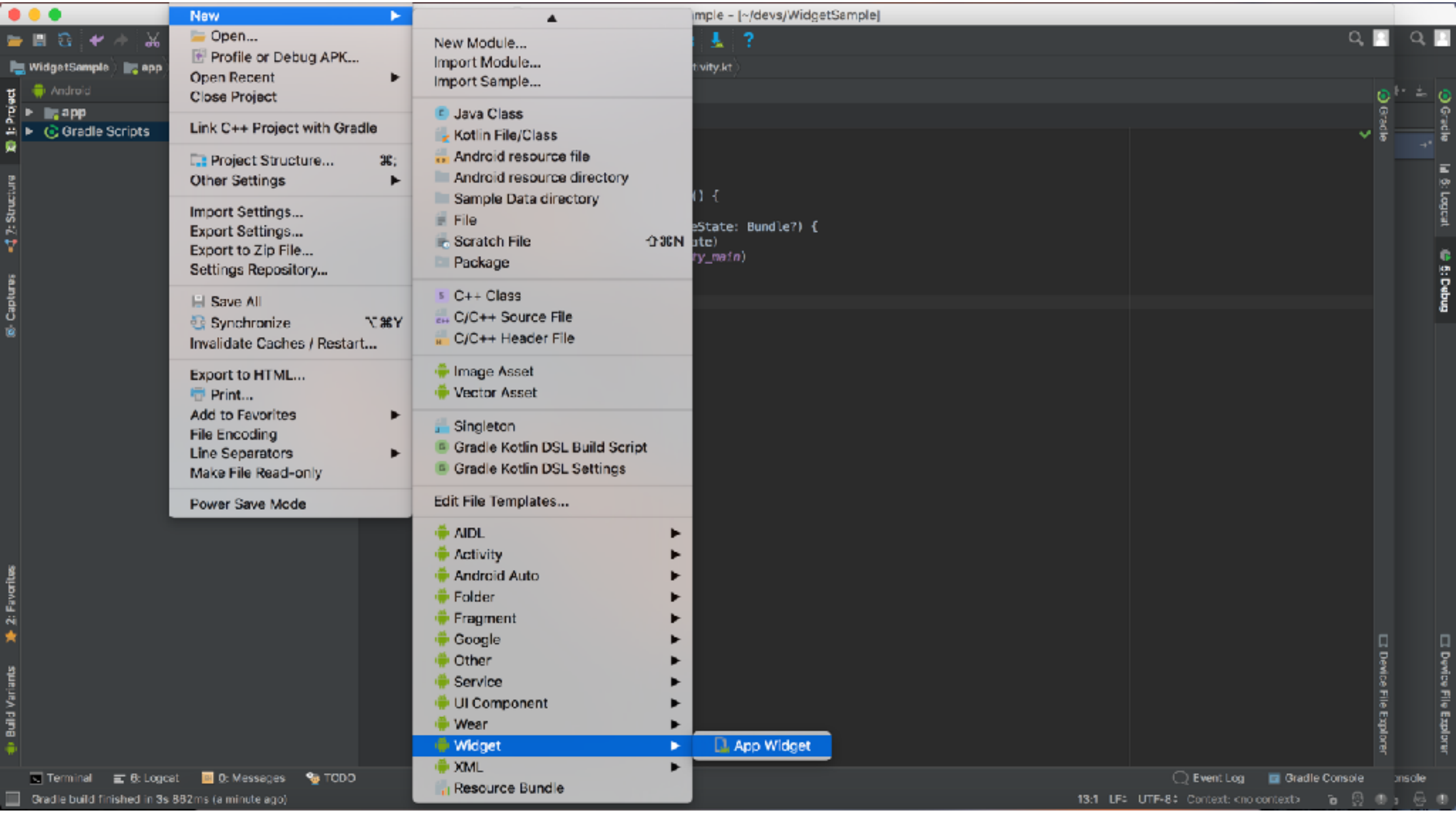
② AppWidgetProviderInfoを設定

③ AppWidgetProviderでRemoteViewを生成＆Update

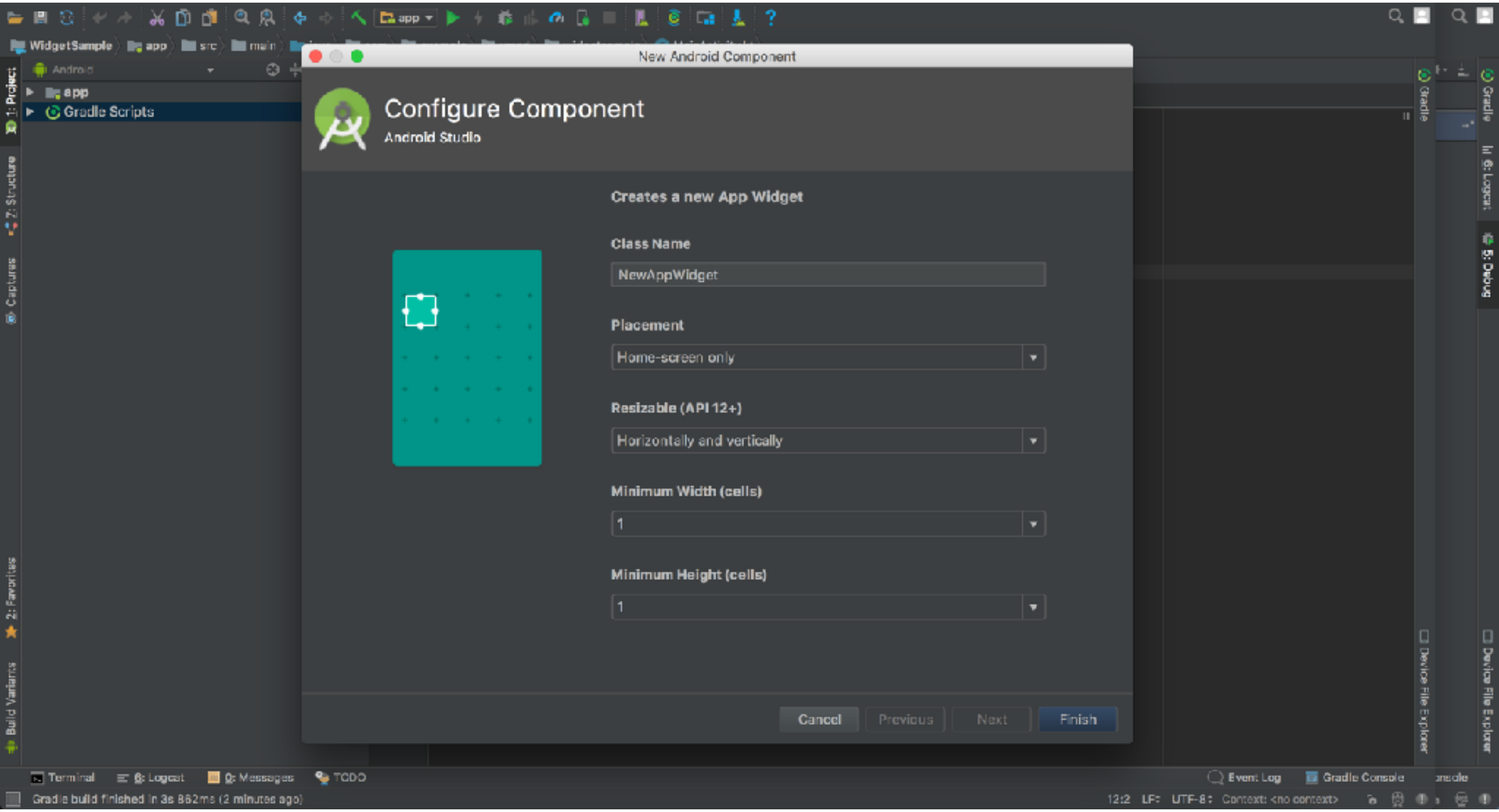
Help me AndroidStudio!!

- AndroidStudioのNewのからウィザードを出せます！！
- Widgetだからリストの一番下（Wだから・・・）

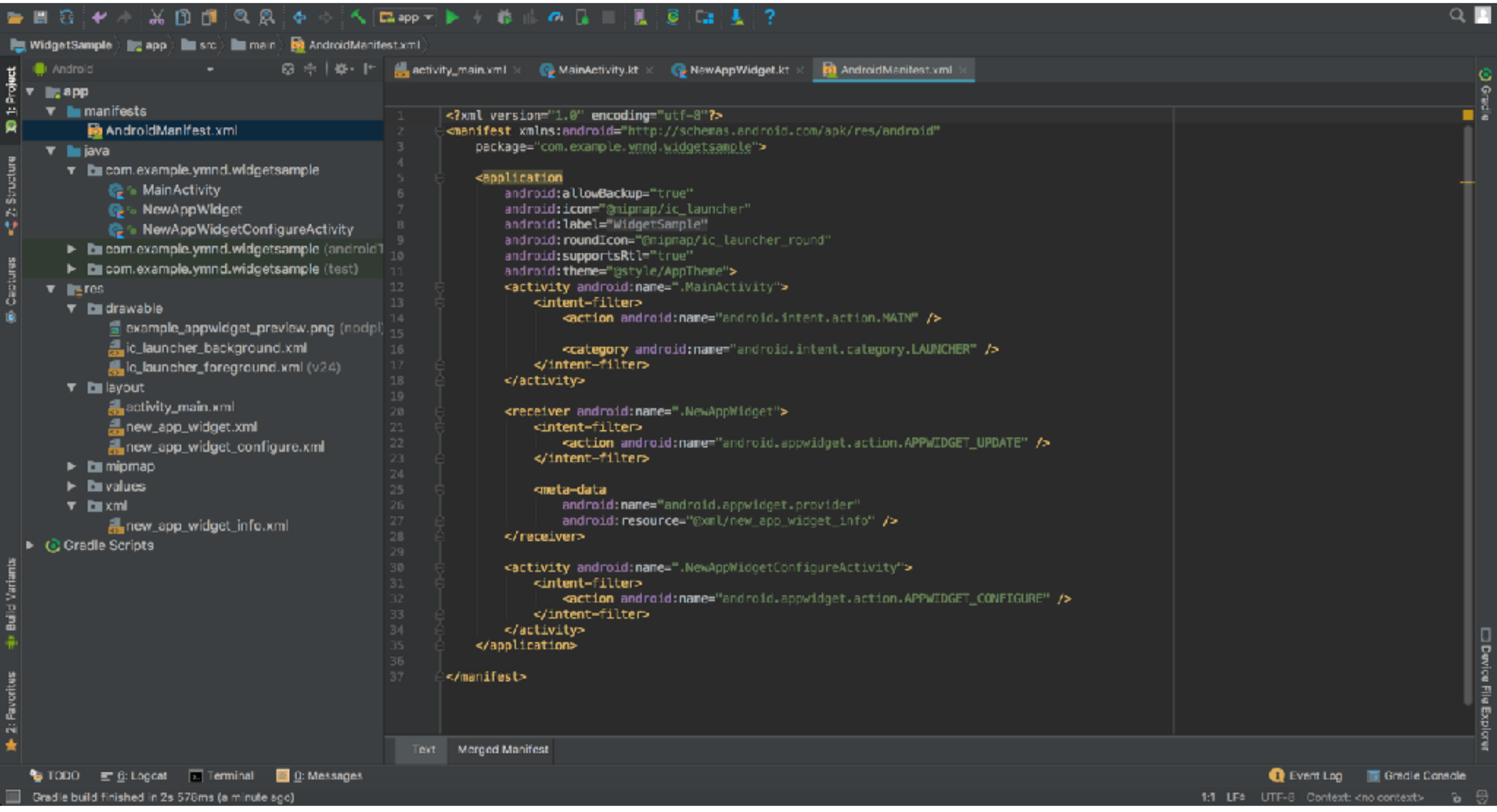
Help me AndroidStudio!!



Help me AndroidStudio!!



Help me AndroidStudio!!



では実際にレイアウトから組み始めよう

WidgetのView

- 基本的に通常のアプリと同様に組むことができます
- Widgetでは、
  - @RemoteViewが付いたViewが使える
  - @RemotableViewMethodが付いたメソッドが使える
- ※@RemoteViews.RemoteViewを使ったとしても

  CustomViewをつくることはできない

Widget: Layout

- FrameLayout
- LinearLayout
- RelativeLayout
- GridLayout
- ※ConstraintLayoutは使えない！！

Widget: 基本要素

- Button
- ImageButton
- ImageView
- ProgressBar
- TextView
- ViewFlipper

Widget: タイマー系

- AnalogClock -> deprecated
- Chronometer -> タイマーをつくる
- TextClock -> デジタル時計を表示させる

Widget: Collection

- ListView
- GridView
- StackView
- AdapterViewFlipper

About RemoteViews

- RemoteViews
- RemoteViewsFactory

RemoteViews

- ビューヒエラルキーを持つクラス
- Widgetの表示の要
- WidgetはServiceにより実装され、Serviceのライフサイクルに従う

RemoteViewsFactory (RemoteViewsService)

- ListViewのItemを作成することができる
- Adapterのような存在

RemoteViewsFactory: Lifecycle//TODO:図示

onCreate(初回作成時に呼ばれる)

↓

onDataSetChanged(DataChangeされたとき)

getViewAt

↓

onDestroy(削除時に呼ばれる)

AppWidgetProvider

- BroadcastReceiverをwrapした便利なクラス
- onReceivedで、イベントをハンドリング
  - Widgetの作成／更新／削除など
- onReceivedをもとに自力で実装も可能です

実装

```
class SampleAppWidgetProvider : AppWidgetProvider() {
    override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager, appWidgetIds: IntArray) {
        for (appWidgetId in appWidgetIds) {
            val intent = Intent(context, MainActivity::class.java)
            val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)
            val views = RemoteViews(context.packageName, R.layout.widget_container)
            views.setTextViewText(R.id.textWidget, "Hello:)")
            views.setOnClickPendingIntent(R.id.button, pendingIntent)
            appWidgetManager.updateAppWidget(appWidgetId, views)
        }
    }
}
```

解説

```
//RemoteViewsに使用したいレイアウトを渡します
val views = RemoteViews(context.packageName,
R.layout.widget_container)
//Textを動的に書き換える
views.setTextViewText(R.id.textWidget, "Hello:)")
//Click時の動作をセットしたいとき
views.setOnClickPendingIntent(R.id.button, pendingIntent)
```

解説: setTextViewText

views.setTextViewText(R.id.textWidget, "Hello") は
setCharSequence(viewId, "setText", text) をwrapしたものなので

views.setInt(viewId, "setBackgroundColor", color)
のようにすれば、RemoteViewに生えてないメソッドも呼べる

実装：ListViewのケース

```
fun updateWidget(context: Context, appWidgetId: Int) {
    val manager = AppWidgetManager.getInstance(context)
    val remoteViews = RemoteViews(context.getPackageName(), R.layout.widget_container)

    // ListItemに更新通知を行うためのintentを作成
    val intent = Intent(context, SampleWidgetService::class.java)
            .putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId)
    remoteViews.setRemoteAdapter(R.id.widgetListView, intent)

    // ListItemが空だった場合のEmpty表示
    remoteViews.setEmptyView(R.id.widgetListView, android.R.id.empty)

    // ListItemをタップしたときのintentをセット
    // ListItemの側でセットするのでなく、ここでセットする
    val onClickIntent = Intent(context,
DetailActivity::class.java).addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
    val onClickPendingIntent = PendingIntent.getActivity(context, 0,
            onClickIntent, PendingIntent.FLAG_UPDATE_CURRENT)
    remoteViews.setPendingIntentTemplate(R.id.widgetListView, onClickPendingIntent)

    manager.updateAppWidget(appWidgetId, remoteViews)
}
```

解説:ListView

```
// ListItemに更新通知を行うためのintentを作成
// 必要があれば情報を付加する
val intent = Intent(context, SampleWidgetService::class.java)
        .putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId)

// 作成したintentはListViewを対象にセット
remoteViews.setRemoteAdapter(R.id.widgetListView, intent)

// ListItemが空だった場合のEmptyViewを指定する
remoteViews.setEmptyView(R.id.widgetListView, android.R.id.empty)
```

解説:ListView

```
// ListItemをタップしたときのintentをセット
// ListItemに紐づくListItem側でセットする
val onClickIntent = Intent(context,
DetailActivity::class.java).addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
val onClickPendingIntent = PendingIntent.getActivity(context, 0,
        onClickIntent, PendingIntent.FLAG_UPDATE_CURRENT)
remoteViews.setPendingIntentTemplate(R.id.widgetListView,
onClickPendingIntent)
```

実装：ListView(Item)のケース

```
override fun getViewAt(position: Int): RemoteViews {
    Log.v("[Widget]", "list getViewAt $position")
    val rv = RemoteViews(context.packageName, R.layout.list_item)
    rv.setTextViewText(R.id.list_text, items[position])
    //ここでIntentを詰める。このintentはこのクラスの呼び出し元へ渡される
    //ListItemの持つデータが欲しい場合はこのintentに詰める
    rv.setOnClickFillInIntent(R.id.list_container, createIntent(position))
    return rv
}
```

AppWidgetProviderのライフサイクル

onEnabled

onUpdated

onDeleted

onDisabled

--- その他 ---

onAppWidgetOptionsChanged

onRestored

onEnabled

- Widgetがはじめてつくられたときに呼ばれる
- ２個置いた場合でも１度きりしか呼ばれない

onUpdated：important

- Widgetの表示を更新する
- onEnabledの直後に呼ばれる
- ２個目からはこれだけしか呼ばれない

onDeleted

- Widgetが消されたときに呼ばれる

onDisabled

- 最後のWidgetが消されたときに呼ばれる

onAppWidgetOptionsChanged

- リサイズされたときに呼ばれる
- 画面サイズをdpで取得できるよ！

onAppWidgetOptionsChanged

- OPTION_MIN_WIDTH：lower bound on the current width
- OPTION_MAX_WIDTH：upper bound on the current width
- OPTION_MIN_HEIGHT：lower bound on the current height
- OPTION_MAX_HEIGHT：upper bound on the current height

onRestored

- added in API level 21
- 直後にonUpdateを呼び出す
- backupからAppWidget providerのインスタンスがリストアされたときに呼ばれる
- widgetに紐付いた永続化データを保持したい場合はここで古いidから新しいidへの引き継ぎを行う
- widgetIdとは？ー＞Appendixへ！

onRecieve

- BroadcastReceiverのonRecieveのこと
- サブクラスではこれをoverrideするときに、各Intentのactionを見て振り分けを行っている
- 振り分けた結果、上のライフサイクルが呼ばれる
- 基本使わなくても済むが、自分でActionを管理したい場合はここを書き換える必要がある

onUpdatedはいつ呼ばれるか

- onEnabledの直後
- onRestoredの直後
- AppWidgetManager.ACTION
- UPDATEが呼ばれたとき

ACTIONAPPWIDGETUPDATEはいつ呼ばれるか

- ウィジェット設置時
- updatePeriodMillisの時間が来たとき
- systemの起動時

updatePeriodMills

- Widgetのアップデート間隔、ProviderInfoで指定する
- 最低更新間隔は
- である（保証はされない）
- ユーザーが更新間隔を調整できると良い
- はここでしている
  - 0以下を指定すると時間更新でbroadcastを投げないようになる
  - 30min未満の場合は、30minにされる
- これより短く更新を行いたい場合は、AlarmManagerなどを使って自分で実装する必要がある
  - この際、updatePeriodMillsは０を指定しておくこと

AppWidgetProviderInfo

- Widgetそのものの設定をxmlで記述する

実装

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minHeight="40dp"
    android:minResizeWidth="250dp"
    android:minWidth="250dp"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="1800000"
    android:configure="com.example.ymnd.widgetsampleapp.ConfigureActivity"
    android:widgetCategory="home_screen|keyguard">
</appwidget-provider>
```

画面の大きさを指定するもの

- int minWidth / int minHeight
  - デフォルトの幅と高さをdpで指定する
- int minResizeWidth / int minResizeHeight
  - リサイズの下限をdpで指定する
    - (minResizeHeight <= minHeightにすること)
- int resizeMode(horizontal|vertical or none)
  - どの方向のリサイズを許可するか

プレビューにかかわるもの

- int previewImage
- Widget Picker／Dialogのプレビューに用いられる

レイアウト

- int initialKeyguardLayout
  - KeyGuardに表示したときのレイアウトを指定する
- int initialLayout
  - ウィジェットのレイアウトを指定する
- int widgetCategory(home_screen|keyguard)
  - ウィジェットをどこに置けるか指定する
  - ※keyguardはAndroid 5.0以上では使用できない

画面更新系

- int updatePeriodMillis
  - ミリ秒でWidgetの更新間隔を指定する
  - 30minより早く更新することはできない
  - 更新をしたくない場合は０を指定すること
- ComponentName configure
  - コンフィグ画面（Activity）をWidget設置時に呼び出すときに使用する
  - ただし、onUpdateは呼ばれないので自分で呼ぶなり画面更新メソッドをたたく必要がある
  - configを設定した場合、home画面にdropして追加した場合に指定したActivityを呼び出してくれる
  - requestDialogから設置したときには呼ばれない
- int autoAdvanceViewId
  - StackViewなど子要素を持つレイアウトのidを指定する
  - ユーザーの操作がなくても自動的に進められる

New API

requestPinAppWidget &
isRequestPinAppWidgetSupported

what is requestPinAppWidget?

- AndroidO追加されたAPI
- アプリの中からWidgetの追加を行えるように！！
- https://android-developers.googleblog.com/2017/07/whats-new-for-shortcuts-and-widgets-in.html

API's

- requestPinAppWidget
  - アプリ内にWidgetを追加する
- isRequestPinAppWidgetSupported
  - ホーム画面に追加できるかチェックする

実装

```
val widgetManager = AppWidgetManager.getInstance(this)
val provider = ComponentName(this, SampleAppWidgetProvider::class.java)
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) return
if (widgetManager.isRequestPinAppWidgetSupported) {
    val callbackIntent = Intent(this, SubActivity::class.java)
    val successCallback
            = PendingIntent.getActivity(this, 0, callbackIntent, 0)
    val extras = Bundle()
    val remoteViews = RemoteViews(this.packageName,
R.layout.widget_container)
    remoteViews.setTextViewText(R.id.button, "てすと")
    extras.putParcelable(EXTRA_APPWIDGET_PREVIEW, remoteViews)
    widgetManager.requestPinAppWidget(provider, extras, null)
}
```

解説: isRequestPinAppWidgetSupported

- DefaultLauncherでWidgetリクエストが使用できるかをチェックする
- requestPinAppWidgetの方でも対応しているかを確認できるがこれを使うことでランチャーがAPIに対応しているかを実行前に確認できる（Appendix）

解説: callbackIntent

- val callbackIntent = Intent(this, SubActivity::class.java)

- 成功時のcallbackで呼びたいintentをここで用意する
- 任意にbundleをセットしてもよい

- 成功時には、AppWidgetManager.EXTRA
- IDがintentに詰め込まれている
- 複数カスタムwidget対応など、widgetの画面で出し分けたい項目があれば、AppWidgetIdを保存しこう

解説: callbackIntent

- Widgetはもともとdropしたときに実行される
- がある
- configはrequestのメソッドでは呼ばれないので、成功時のconfig／アプリへの復帰はここでセットする

解説: successCallback

- val successCallback = PendingIntent.getActivity(this, 0, callbackIntent, 0)
- Widgetをおいたあとに実行するcallbackをPendingIntentとしてセットする
- Widgetの追加ダイアログは、自動で追加or手動で追加の２パターン存在が存在する
- 自動的に追加の場合はアプリの画面から遷移しないのでよいが、手動だとホームランチャーに遷移
- 自分のアプリに戻ってこないので、Broadcastで通知するなりActivityを再度開くなりで登録を行う
- また、設定画面を引っ掛けることでユーザーにカスタムWidgetとして使ってもらうこともできる

解説: extras

```
val extras = Bundle()
val remoteViews = RemoteViews(this.packageName,
R.layout.widget_container)
remoteViews.setTextViewText(R.id.button, "てすと")
extras.putParcelable(EXTRA_APPWIDGET_PREVIEW, remoteViews)
```

- このbundleによりcustom previewをセットできる
- previewImageと異なり、remoteViewsをカスタムしてセット可能である
- ただ、preview用であり、onUpdateされる際に更新さ

解説: extras

- このbundleによりcustom previewをセットできる
- previewImageと異なり、remoteViewsをカスタムしてセット可能である
- ただ、preview用であり、onUpdateされる際に更新されてしまう

解説: requestPinAppWidget

- WidgetDialogを表示するメソッド
- 返り値はbooleanである
- trueの場合、requestPinAppWidgetに対応している意味である
- これはWidgetが追加できたことを意味しない

requestPinAppWidgetでWidgetの配置に失敗したらどうなるか

- requestPinAppWidgetはWidgetの作成成功時にしか successCallbackを呼ばない
- fallbackがないので、手動配置で失敗した場合、アプリを再び開いてもらう必要がある

requestの注意点

- 追加ダイアログは自分でカスタムすることはできない
- Previewを変えることはできるー＞extrasにremoteViewを詰めてあげてね！！

Conclusion

- Widgetのダイアログにより画期的に導線が分かりやすくなった
- 新規APIを含め実装が簡単なので気軽につくりはじめられる
- 多くのアプリが端末にインストールされている今だからホームの特等地をとれるWidgetを！！

whoami

- twitter:@ymnd, github:@ymnder
- Application Engineer
  - Android日経電子版アプリ
  - Android紙面ビューアーアプリ
- 技術書典４で日経電子版の知見をまとめた本を出します。

Appendix

How to Calculate minWidth and minHeight

WidgetのminWidth/minHeightは如何にして算出するか？

算出方法

70 × cell − 30 = size

例：1行2列のwidgetを作成する場合は

minHeight = 70 x 1 - 30 = 40

minWidth = 70 x 2 -30 = 110

How to Calculate minWidth and minHeight

- API14以前で推奨されていた計算方法がアップデートされた！
- targetSDKがAPI14以上の場合、ホーム画面のウィジェットの各端にマージンが自動的に追加される
- そのため、Android4.0以上ではwidgetに予め余白を追加する必要はない

Config Best Practice(from Material Design Guideline)

- configは簡潔にし、２〜３以上の設定項目を設けない
- コンテクストを考えるとフルスクリーンよりダイアログで選択してもらう
- セットアップ後はWidgetには設定用のボタンを表示しない

how is appWidgetId created?

- appWidgetIdはWidgetを更新するときに用いるID
- ユニークであるはずだけど、このIDがどのように振られているかが気になる

intentにはこのKeyで詰められている

```
AppWidgetManager#EXTRA_APPWIDGET_ID
    An intent extra that contains one appWidgetId.
```

AppWidgetHost.java

```
//Get a appWidgetId for a host in the calling process.
//@return a appWidgetId
public int allocateAppWidgetId() {...}
```

http://tools.oesf.biz/android-8.0.0_r1.0/xref/frameworks/base/core/java/android/appwidget/AppWidgetHost.java#allocateAppWidgetId

AppWidgetServiceImpl.java

```
public int allocateAppWidgetId(String callingPackage, int hostId) {
...
if (mNextAppWidgetIds.indexOfKey(userId) < 0) {
    // 初期化処理
    // AppWidgetManager.INVALID_APPWIDGET_IDは0
    // Widgetは1から採番されていく。
    mNextAppWidgetIds.put(userId, AppWidgetManager.INVALID_APPWIDGET_ID + 1);
}
final int appWidgetId = incrementAndGetAppWidgetIdLocked(userId);
...
}
```

http://tools.oesf.biz/android-8.0.0_r1.0/xref/frameworks/base/services/appwidget/java/com/android/server/appwidget/AppWidgetServiceImpl.java#allocateAppWidgetId

どのように番号が振られるか？

```
private final SparseIntArray mNextAppWidgetIds = new
SparseIntArray();
final int appWidgetId = peekNextAppWidgetIdLocked(userId) + 1;
return mNextAppWidgetIds.get(userId);
```

mNextAppWidgetIdsには番号が格納されている

これは、{userId, appWidgetId}のKey,Valueで保存される

読み出した後、+1して格納するという

インクリメント処理
が行われている

どのように番号が振られるか？

- 結論：特別な値を生成しているわけではない。
- AppWidgetServiceImplはServiceクラスであり、他のアプリからもアクセスされる
- mNextAppWidgetIdsは他のアプリと共有されている
- 取得したい場合はこれを使えば取得できる

requestPinAppWidget & isRequestPinAppWidgetSupportedの返り値

- isRequestPinAppWidgetSupported
  - Return {@code TRUE} if the default launcher supports
- updateAppWidget
- {@code TRUE} if the launcher supports this feature. Note the API will return without
  - waiting for the user to respond, so getting {@code TRUE} from this API does
  - mean
  - the shortcut is pinned. {@code FALSE} if the launcher doesn't support this feature.
- それぞれbooleanを返すが、これにはどんな違いがあるのか

isRequestPinAppWidgetSupportedの返り値

- isRequestPinAppWidgetSupported
- ShortcutService.java#getRequestPinConfirmationActivity

ShortcutService.java#getRequestPinConfirmationActivity

- ここでdefault launcherから機能がハンドルされている Activityを探している
- 取得できなかった場合はnullを返す
- これがnullであればfalseを返しており、これが非対応という判定を下している。

requestPinAppWidgetの返り値

- AppWidgetManager
- AppWidgetServiceImpl
- ShortcutService#requestPinAppWidget
- ShortcutService.java#requestPinItem
- ShortcutRequestPinProcessor.java#requestPinItemLocked
- ShortcutService.java#getRequestPinConfirmationActivity

requestPinAppWidgetの返り値

- ここでdefault launcherから機能がハンドルされている Activityを探している
- 結果的にisRequestPinAppWidgetSupportedと同じ判定を行っている
- ただ、もしtrueだった場合はそのままrequest処理が進んでいく

返り値の差異

- 同じものである
- リクエスト前に判定をすべきなので、

  isRequestPinAppWidgetSupportedを使う

- docの通り、requestPinAppWidgetは、Widgetを置けたかどうかを返さない

requestPinAppWidgetのcallbackはどこで呼ばれるか？

- 何をもってsuccessと判定しているのだろうか。
- またfallbackとして何か呼べたりはしないだろうか。

callbackはどこで呼ばれるか？

- AppWidgetManager
- AppWidgetServiceImpl
- ShortcutService#requestPinAppWidget
- ShortcutService.java#requestPinItem

ShortcutService.java#requestPinItem

- このメソッドはShortcutの方からも呼ばれている
- //
  ShortcutRequestPinProcessor.java#requestPinItemLocked
- requestPinItemLocked(ShortcutInfo inShortcut,
- AppWidgetProviderInfo inAppWidget,
- Bundle extras,
- int userId,
- IntentSender resultIntent)

callbackはどのタイミングで呼ばれるか？

- 成功時にacceptを呼ぶ
- ここでacceptの中身が実装されている
- 終点：例外をキャッチしていて、作成できなかった場合はfalseを返す：tryAcceptがfalseになり、コールバックが実行されない

callbackはどのタイミングで呼ばれるか？（終点）

```
// PinItemRequestInner#accept
// Pin it and send the result intent.
if (tryAccept()) {
    mProcessor.sendResultIntent(mResultIntent, extras);
    return true;
} else {
    return false;
}
```

callbackはどのタイミングで呼ばれるか？（終点）

- tryAcceptがtrueであったときのみsendResultIntentが実行される
- すなわちここでcallbackが実行される
- 失敗時は何もされない。falseが返るだけ。

Further reading

- 是非ご一読ください:))
- https://developer.android.com/guide/topics/appwidgets/index.html

Widgetやっていき💪