本日の内容

Intent

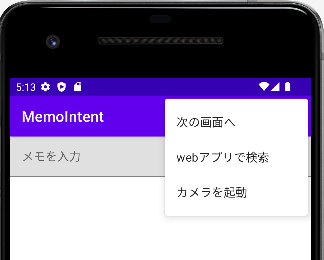
オーバーフローメニュー

今回マニュアル感がとても強いです。

Androidアプリ内で出来ることを増やす目的で授業を行います。

オーバーフローメニュー(公式ドキュメント)

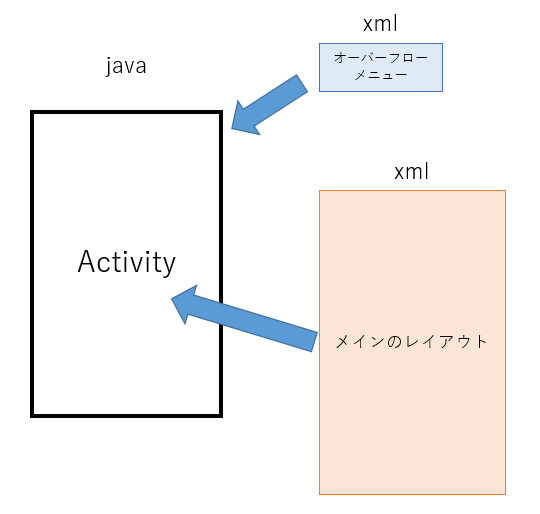
<https://developer.android.com/guide/topics/ui/menus?hl=ja>



オーバーフローメニューとは

Androidアプリ内に作成する、メニューボタンのようなものです。

使用する場合このオーバーフローメニューに対してもレイアウトファイルが必要になります。



■プロジェクト作成

- Empty Activity

- プロジェクト名：MemoIntent

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

今回、メモアプリを作成して学んでいきますが

「文言」についても先に触れておきます。

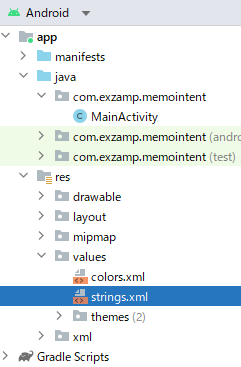
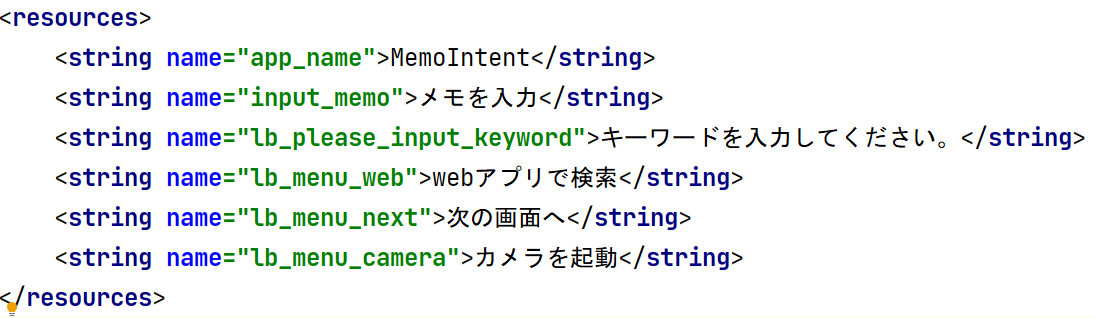
表示文字などは基本的にプログラムを書き換えて変更するようにするのではなく

文字列を定義したファイルを書き換えて変更するように作成するのが一般的です。

何故、プログラム内で変更でなく、ファイルで変更する方が良いのか考えてみましょう。

■①string.xmlに使用する文言を登録

文言が定義されているファイルはres>valuesフォルダの中に配置されています。



上記のように、今回使用する文言を定義しておきましょう。

定義方法は

<string name=”定義名”> 値 </string>

app\_nameを見ると、アプリ名はここで定義されていたことがわかります

■②オーバーフローメニューを作成

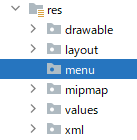
オーバーフローメニューのレイアウトを作成する際、layoutフォルダではなく

リソースフォルダと呼ばれる特別な領域から読み込むことになるので

まずはリソースフォルダのmenuフォルダ作成し、その中にオーバーフローメニューのxmlを作成しましょう。

resフォルダを右クリック>new>Android Resouce Directoryを選択

ResouceTypeをmenuに変更>OKを選択

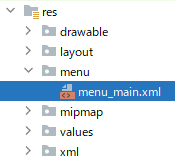


リソースフォルダのmenuが作成されました。

オーバーフローメニュー用のmenu\_main.xmlを作成

Menuフォルダを右クリック>New>XML>Layout XML

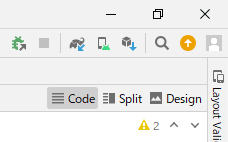
LayoutFileName : menu\_main



生成したxmlファイルをオーバーフローメニュー用に書き換えていきます。

Menu\_main.xmlを開きましょう

Codeタブをクリックし、ソースコードから変更していきましょう。



linearLayoutの作りになっているので

下記のようにルートタグをmenuに書き換えます。

予測変換を活用し、打ち間違いが無いようにしましょう。



これにより、メニューの領域を作成したので

項目を追加しましょう。



・Order In Categoryの値はメニュー項目が表示される順序を定義しています、(昇順)

・showAsActionの値はツールバー上に項目をどのように表示されるかを定義しています。

|  |  |
| --- | --- |
| ifRoom | 配置が許される範囲で表示 |
| always | 常に表示 |
| never | 常に表示しない |

今回はツールバー上に表示させたくない為、neverを指定しています。

デザインタブで現状を確認してみると、メニューに項目が増えていますね



では更に２つのメニューを追加しましょう



追加後、デザインタブで確認してみます。



計３つのメニューが作成出来ました。

メイン画面に対して画面を設計していきます。

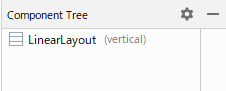
まずはデフォルトであったTextViewを削除します。

次にテキスト入力エリアを配置しましょう

・ activity\_main.xml

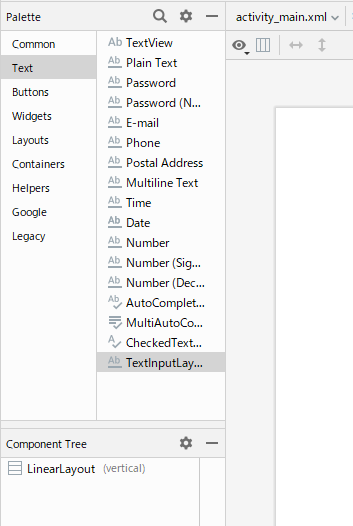
- ルートタグをlinearLayoutに変更

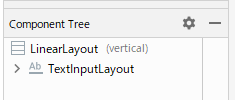
- orientation(向き)をvarticalに変更



続いて、入力エリアのウィジェットを追加します。

- TextInputLayoutを配置



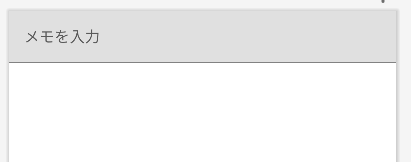
Hintと書かれている部分は入力項目の説明内容になります。

ここには、string.xmlで定義した内容を表示させましょう！

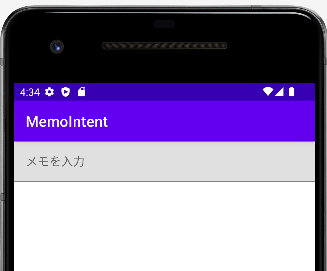
- hint属性を@string/hint\_memoに変更

- idをetInputに変更

変更後、表示されるテキストがstring.xmlにて定義した内容が表示されるようになりました



この状態でアプリを実行してみましょう！



作成したはずのオーバーフローメニューが表示されていません・・・。

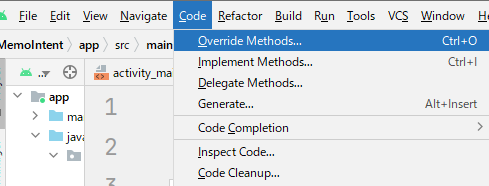
これは、このActivity内で「オーバーフローメニューを配置する」という処理を追記していない為です。

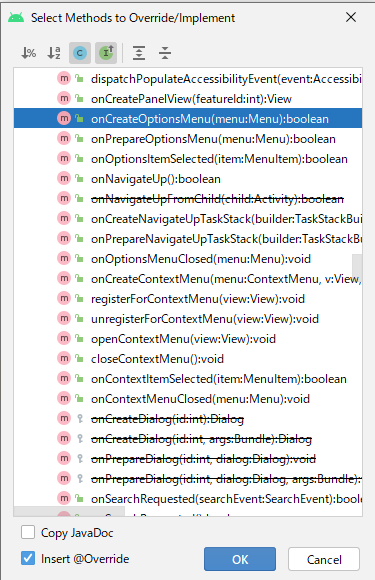
なので、オーバーフローメニューを表示する処理を追記していきましょう！

・MainActivity.java

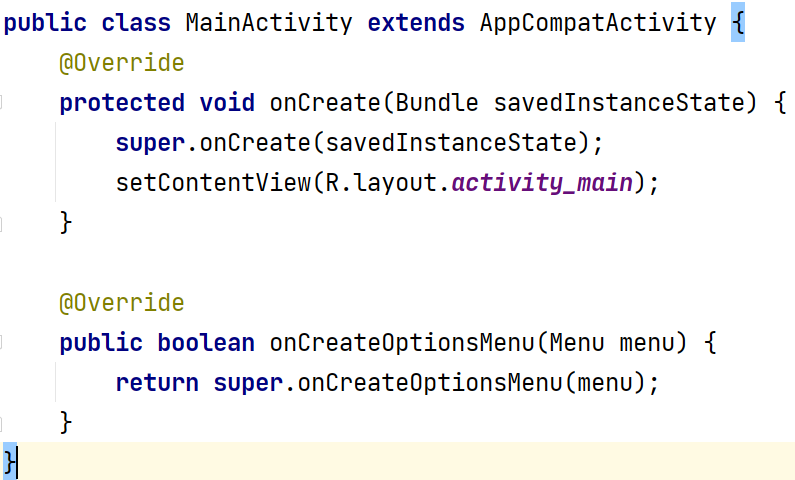
* オーバーフローメニューを表示

Code>Override Methodsをクリック> onCreateOptionsMenu > OK





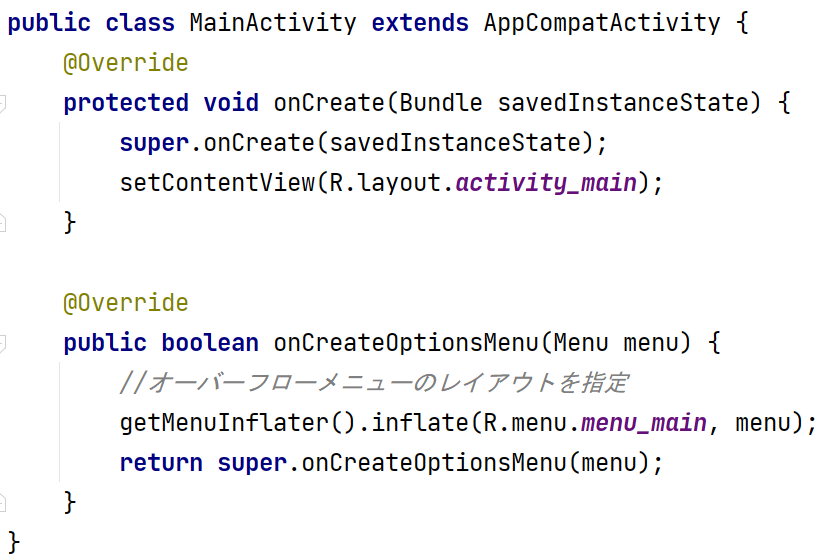
すると、自動的にオーバーライドした、オーバーフローメニュー作成用のメソッドが実装されます。



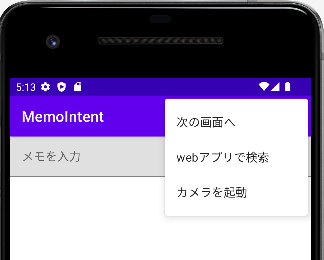
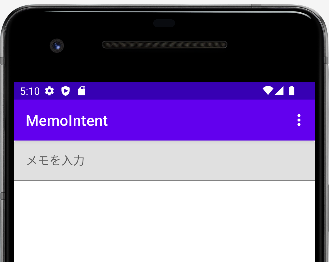
デフォルトのままでは、読み込むファイルを指定していませんので、

作成したmain\_menu.xmlをオーバーフローメニューとして読み込むように処理を追加しましょう。

メソッド名は「onCreateOptionsMenu」



改めてアプリを実行してみましょう



オーバーフローメニューが追加され、リストが表示されるようになりました！

しかしこのままでは特に処理がありません。

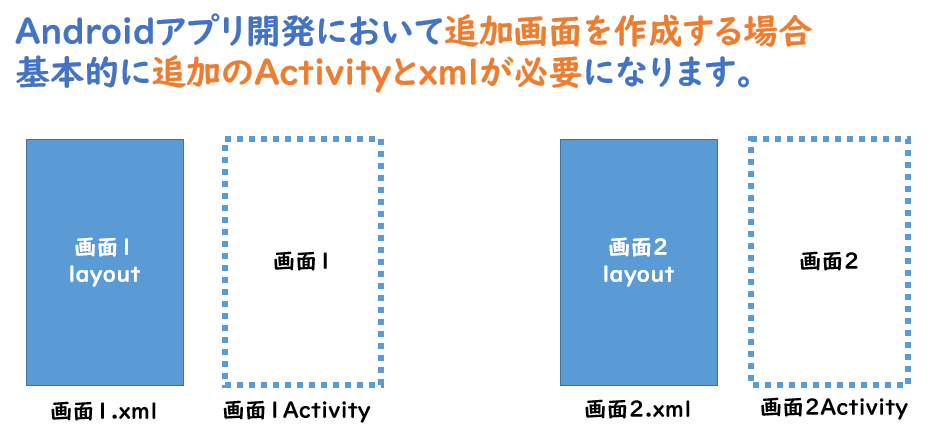
次は、「次の画面へ」を押した場合、別画面に移動するような処理を作りたいので

遷移する画面を作成していきましょう！

■③：1画面と1Activity

Androidでは、複数の画面を１つのファイルで構成しているのではなく、

基本的に1画面に対して1Activityで構成されます。



2つ目の画面の為にActivityを作成していきましょう

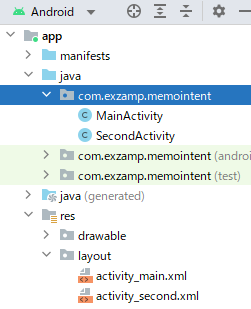
・SecondActivity.java

* 作成

MainActivityがあるパッケージを右クリック>new>Activity>Empty Activityをクリック

activityNameを「SecondActivity」と入力>Finish

するとActivityとxmlが生成されます。

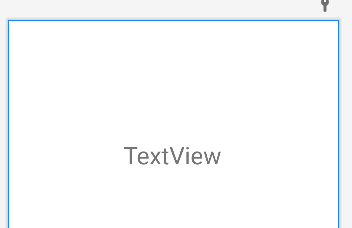


Manifest.xmlも確認すると、使用するActivityとしてSecondActivityが追記されています。



次の画面のレイアウトも編集しておきましょう。

・activity\_second.xml

* TextViewを配置
* textSizeを30spに変更
* idをtvMsgに変更
* 

では、作成した画面に遷移するように処理を追加していきます

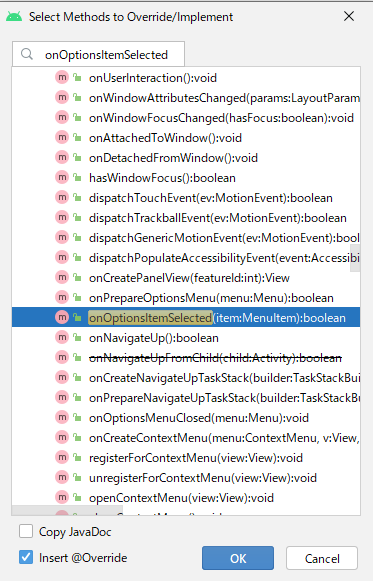
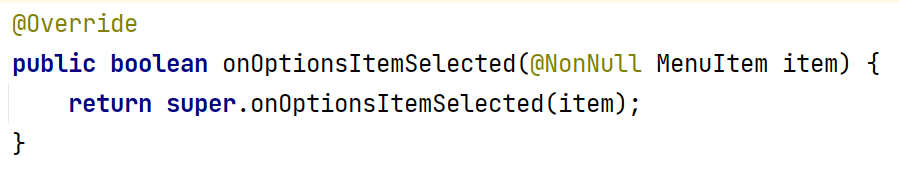
・MainActivity.java

オーバーフローメニューのクリック処理

どのオーバーフローメニューが押されたか検知する為のメソッドも用意されているので

オーバーライドで実装しましょう

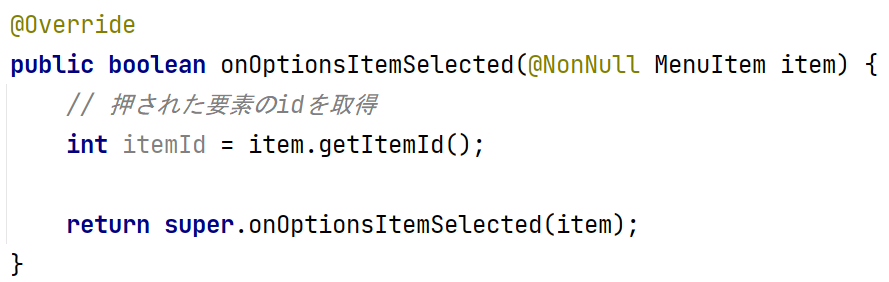
メソッド名は「onOptioinsItemSelected」



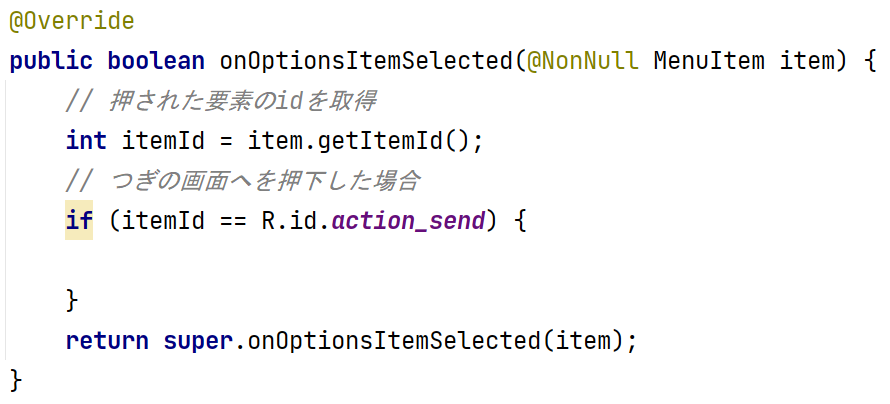
このonOptioinsItemSelectedの引数itemに押された要素のオブジェクトが格納されています。

このitemを使用しどの要素を押下したのか判別できるようにしましょう。

item.getItem()メソッドを使用する事で、押下した要素のidを取得します。

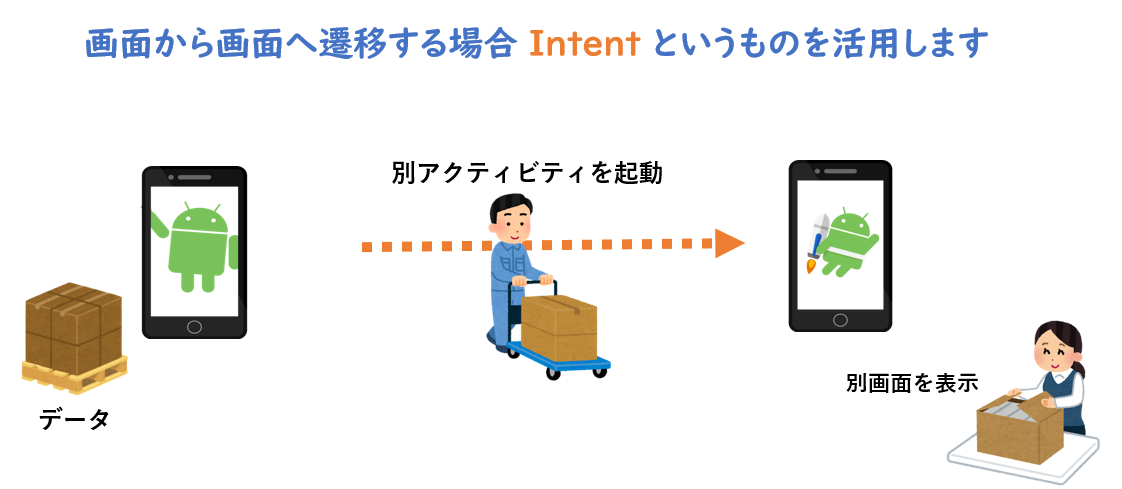


取得したidで処理を分岐するように処理を追記しましょう。



これで「次の画面へ」を押された際にだけ処理を行う準備が整いました。

* ④：画面遷移



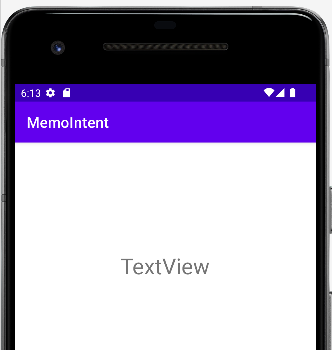
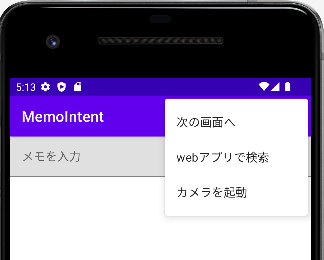
Intentのオブジェクトは移動用の箱のようなイメージで大丈夫です。

Intentは、「何が」「どこへ移動するのか」の情報を元に作成します。

startActivityで次の画面へ向けて出発します。



この状態で再度アプリを実行してみましょう！



画面遷移を行うことが出来ました！

Intentではただ、画面を遷移するだけでなく、

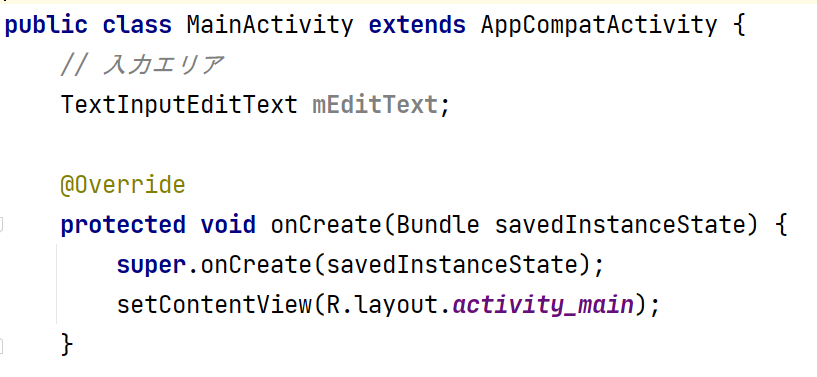
データを次の画面へ送信することも可能です。

入力された文字を次の画面でも表示出来るように追記していきましょう。

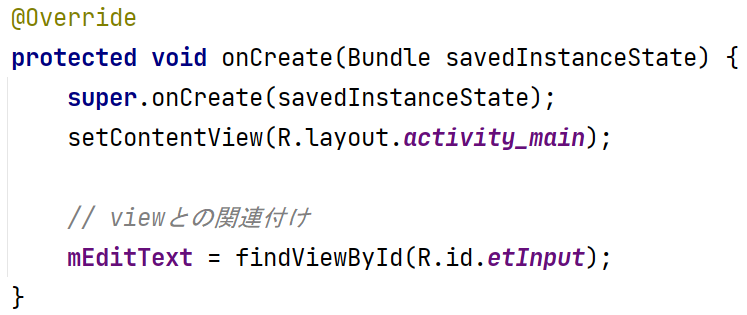
* 入力されているテキストを取得

まずは入力エリアである、TextInputEditTextの変数をフィールドに宣言します。

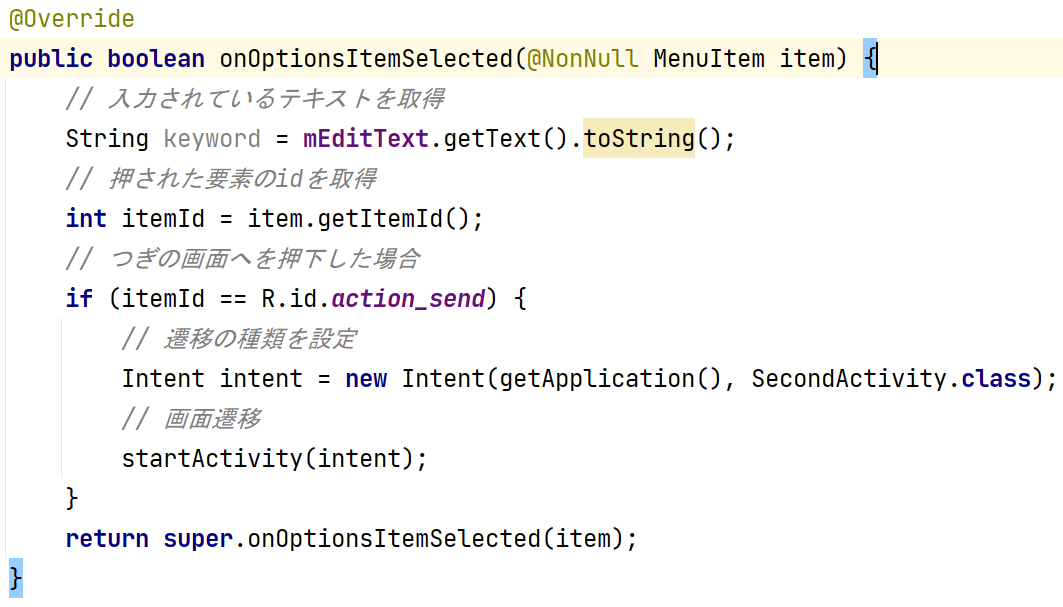
* ⑤：TextInputEditTextの宣言



* ⑥：xmlのTextInputEditTextと関連付け



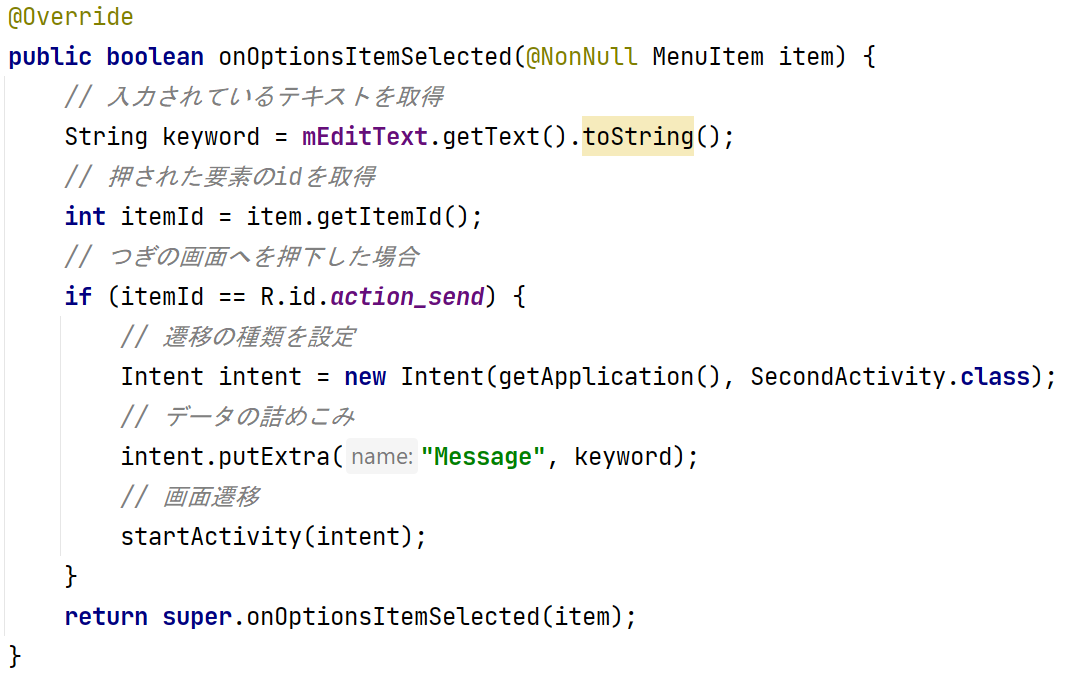
* ⑦入力されている文字の取得



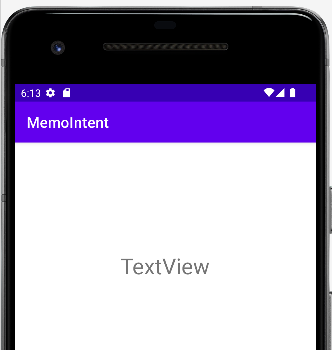
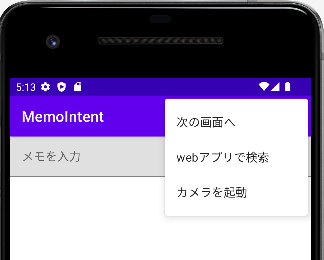
* ⑧データの詰めこみ

データを詰めこむ場合は、IntentのputExtraメソッドを使用します

intent.putExtra(データのラベル, 送信するデータ)



この状態で実行してみましょう！

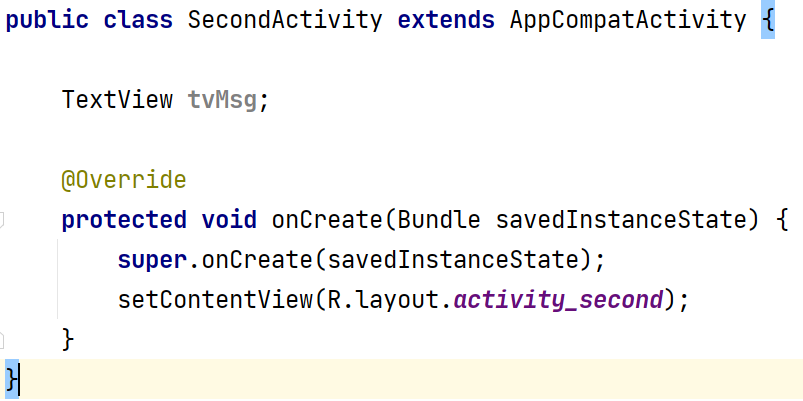
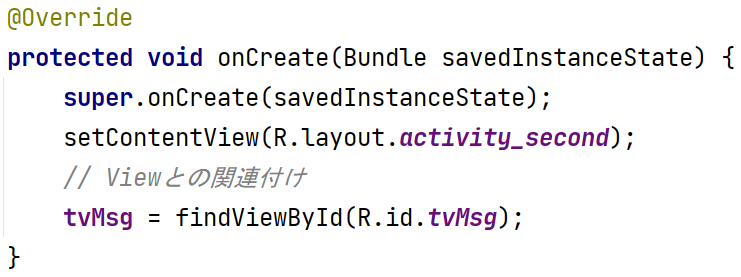


何も変わりません。。。

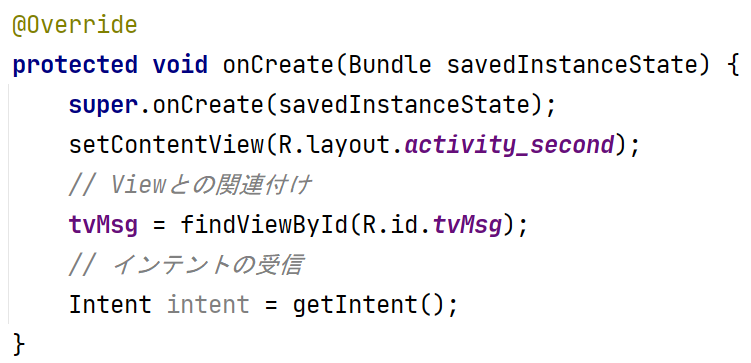
何故なら受け取り側で何も処理を行っていないからです。

次に、データを受け取り、TextViewに反映させる処理を追加していきましょう。

・SecondActivity.java

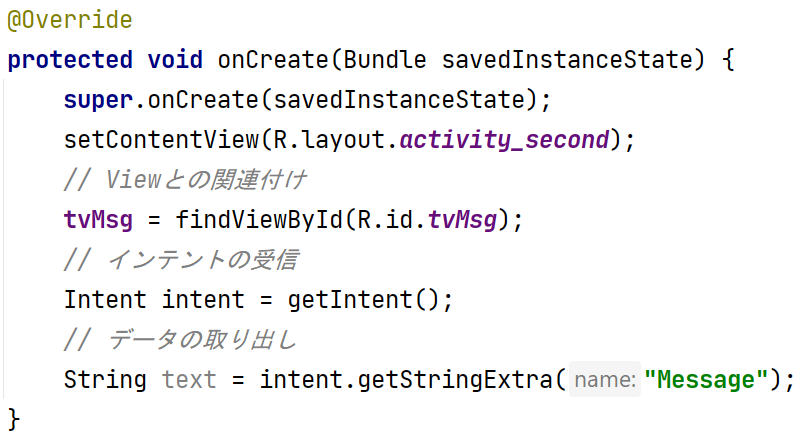
* TextViewの変数定義
* 
* Viewとの関連付け
* 
* インテントの受信

データを受け取るためにはまずはintentを取得する必要があります。



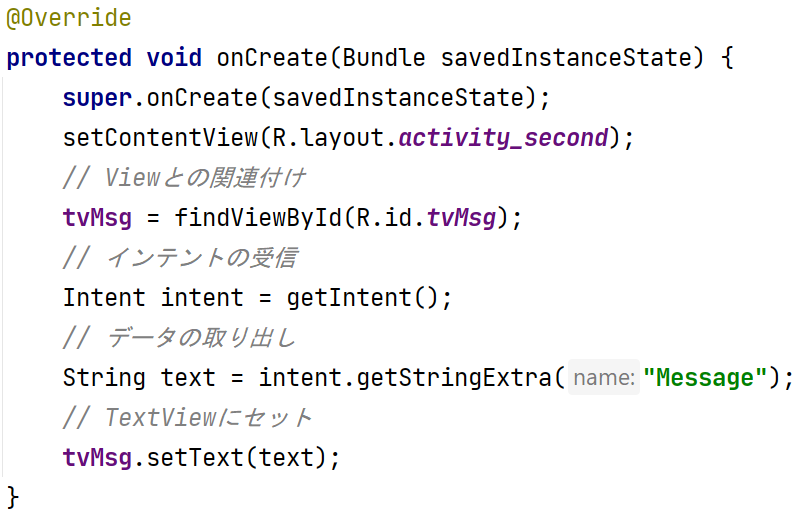
* データの取り出し

Intentを取得した上で、中身を取り出します。



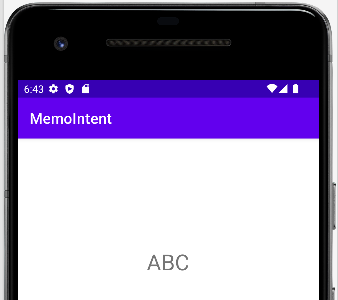
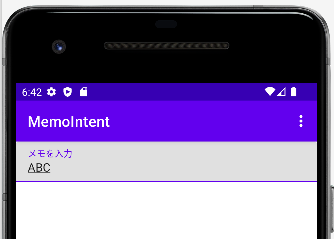
* TextViewにセット

取得した文字列をTextViewにセットし表示させる処理を追加しましょう。



この状態で再度、実行してみましょう

入力エリアに文字を入れた状態で画面遷移

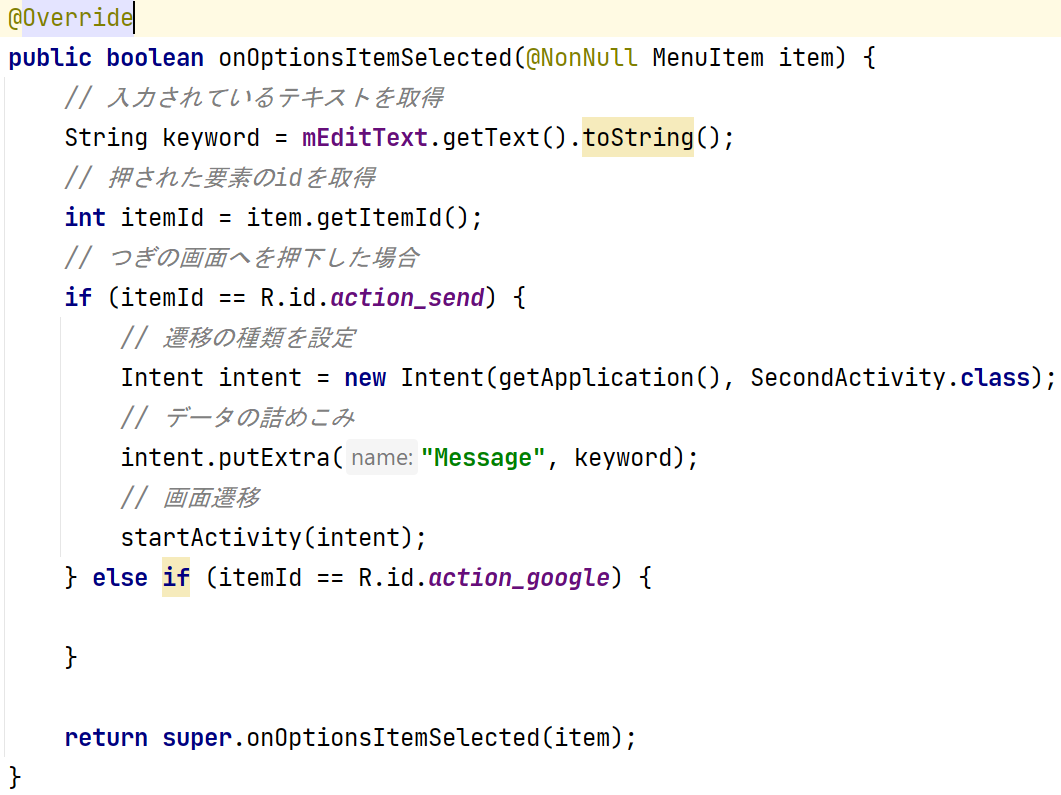


画面遷移では、アプリ内の画面だけでなく、別アプリを起動する画面遷移も可能です！

今回はWebアプリを起動し、検索も行う処理を追加してみましょう！

・Webアプリで検索

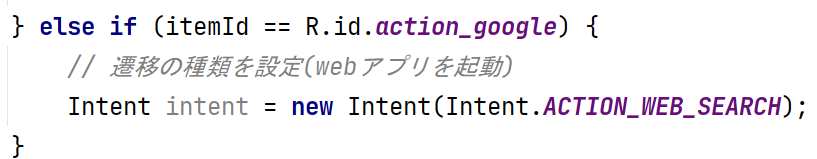
「webアプリを起動」が押されたら分岐するよう処理を追記



* ⑨遷移の種類を設定(webアプリを起動)

アプリを起動する場合は、Intent.起動するカテゴリで指定できます。

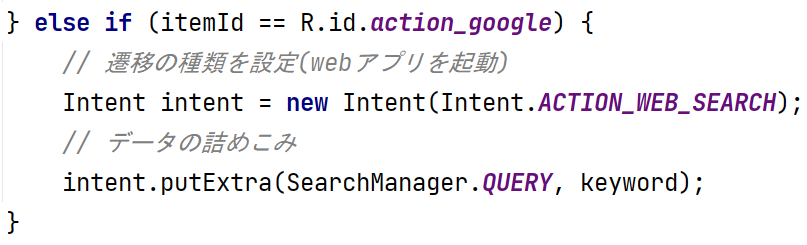
他にもメールアプリの起動や電話アプリの起動などあります。



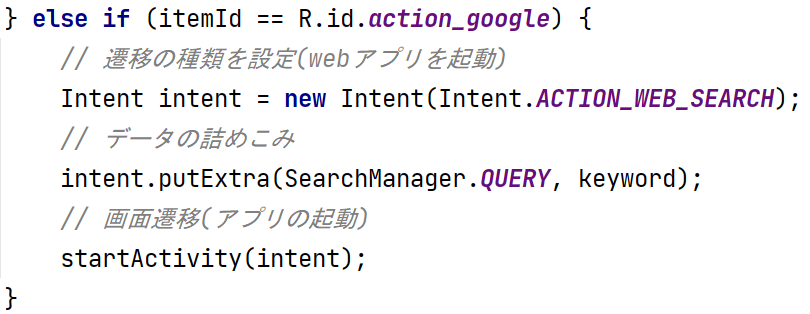
* ⑩データの詰めこみ

今回のラベルでは、受け取りアプリ側が検索用のデータと認識出来るように記述します。

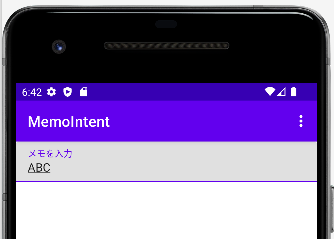
このラベル部分はアプリによって異なるため、それぞれ調べる必要があります。



* ⑪画面遷移(アプリの起動)



この状態で実行してみましょう！

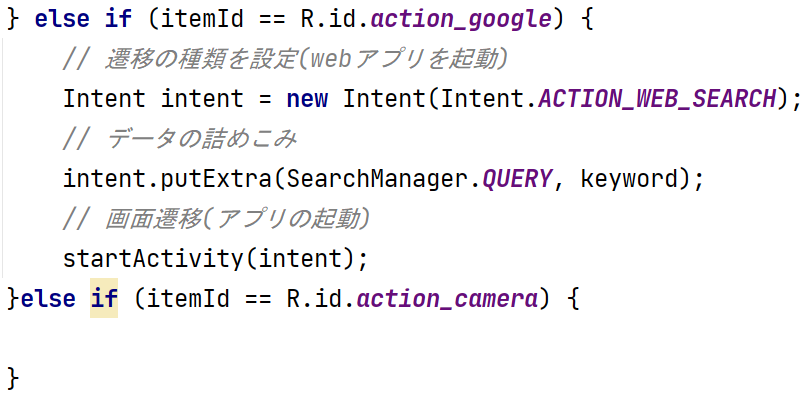


ただし、これはwebアプリを起動しただけで、アプリ内でwebサイトを開いた訳ではありません。

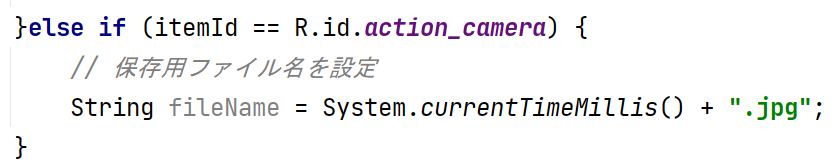
**■ここからex**

・カメラを起動し画像をメモ

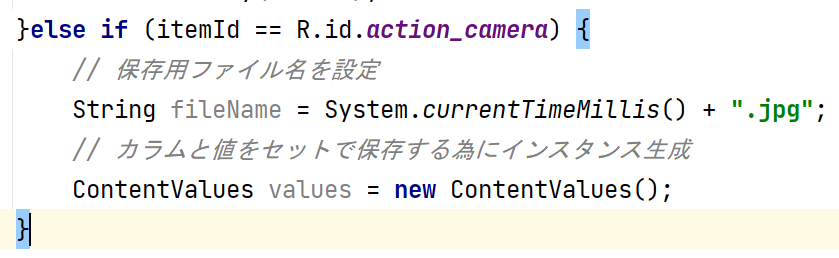
「カメラを起動」をクリックした際の分岐処理



* 保存用ファイル名を設定

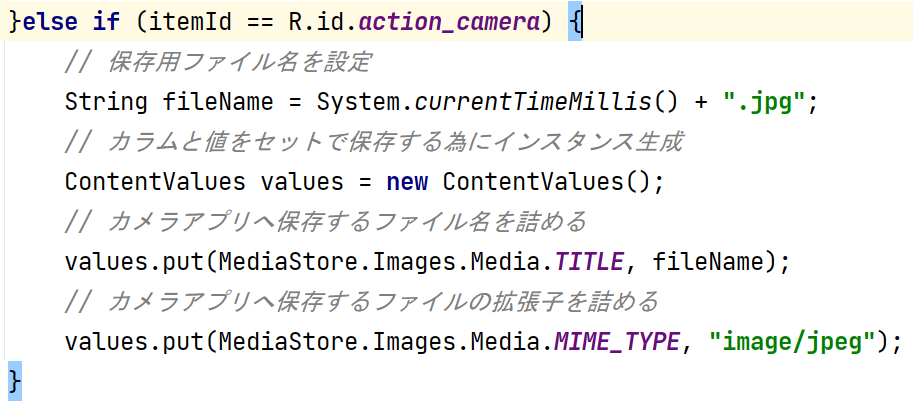


* カラムと値をセットで保存する為にインスタンス生成

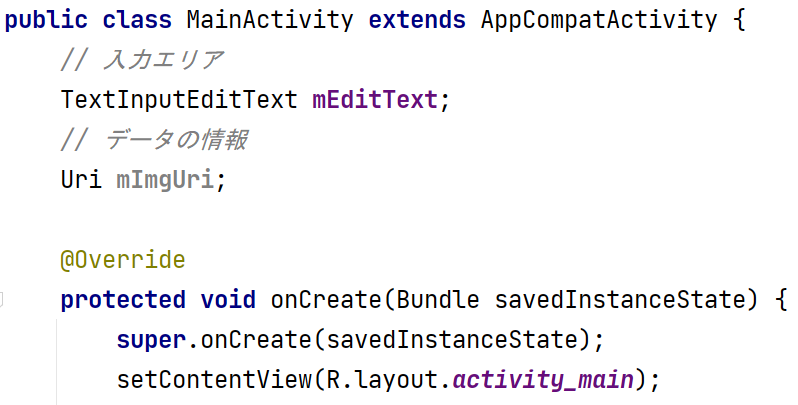


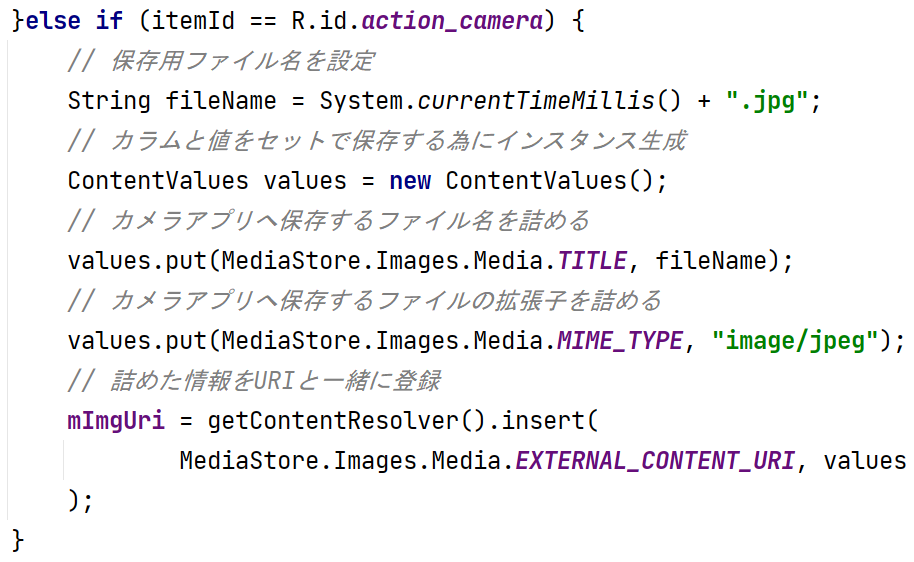
* カメラアプリへ保存するファイル名を詰める

&カメラアプリへ保存するファイルの拡張子を詰める



* 詰めた情報をURIと一緒に登録





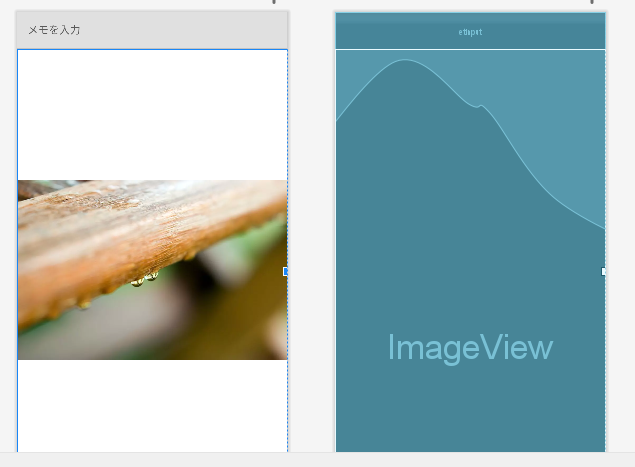
* 画面遷移の設定&カメラアプリへ渡すデータを詰める



■カメラで撮った画像を受け取り、表示するように

Activity\_main.xmlにImageViewを配置 (初期画像は適当)

-idをivPicに設定

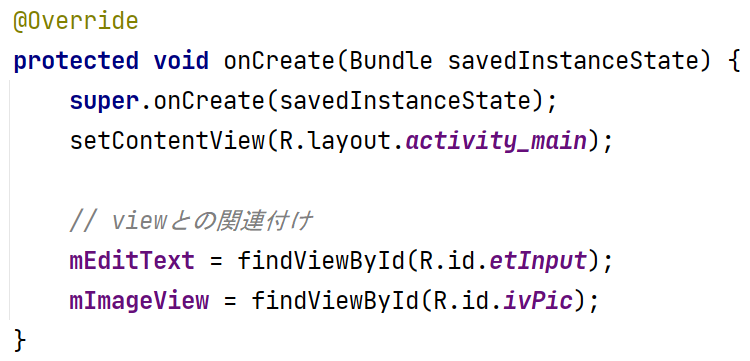


* 戻ってくる前提の画面遷移開始

撮影した写真を表示するようのImageViewの変数を定義



Viewとの関連付け



戻ってきた際に、画像ファイルをImageViewにセットするよう処理を追加



戻ってくる前提の画面遷移を行う

