BlasAndroidアプリ開発における推奨ライブラリについて

fukuda atushi

目次

5. View関連
    5.1. Groupie

6. アーキテクチャについて
    6.1. MVVM

7. Jetpackの利用
    7.1. DataBinding
    7.2. ViewModelとDataBindingについて
    7.3. Roomについて

1. 概要

推奨ライブラリについて

現状における開発環境を確認したところ、Web通信、非同期処理などで使えるライブラリを導入する余地があると考え、各種ライブラリの紹介と、メリットなどについて記載します。

アーキテクチャについて

アーキテクチャの主な導入理由については、フォルダ構成の構築と関心の分離が主な目的であり、
各クラスで単一責務を原則として、コードの肥大化と煩雑さを解消するのが狙い。

AndroidJetpackについて

アーキテクチャコンポーネントなど含むライブラリコレクション。
ライフサイクルを管理したり、色々と便利。
ViewModel,Databindingなどは画面周りのロジックを分離するのに必要。

2. Web通信

Retrofit

https://github.com/square/retrofit
WebAPIコールを比較的かんたんに実装できるライブラリ。
Jsonコンバーターと併せて使用する。
使い方はWebサイトにて検索すれば大量ヒットする。

OkHttp

https://github.com/square/okhttp
Httpクライアントのためのライブラリ。
Http接続部分のコードを簡略化できる。

HttpURLConnectionを使った場合との比較
https://qiita.com/LyricalMaestro0/items/698c77f5a964b5658bbb

MoshiConverter

---

https://github.com/square/retrofit/tree/master/retrofit-converters/moshi
Jsonをパースするためのライブラリ。
Kotlinのdata classを使えば便利。
data classのプロパティをjsonレスポンスのフィールド名に合わせれば、
意識せずフィールド↔プロパティの変換ができる。

3. 画像表示

Glide

https://github.com/bumptech/glide
画像表示ライブラリ。
Web、ローカルどちらの画像も表示でき、リモートから取得した画像は自動的にキャッシュされる。
キャッシュサイズも変更可能である。
オフラインの場合は、キャッシュした画像を自動的に表示することが可能。

Picasso

https://github.com/square/picasso
Glideと同じ様な機能を揃えているが、最近はライブラリ自体あまり更新されておらず、
Glideを使ったほうが無難。

4. 非同期処理

RxJava

https://github.com/ReactiveX/RxJava
Reactive ExtensionsなプログラミングをJavaでも出来るよう開発されたライブラリ。
オブザーバーパターンなどを参考にしてもらえると分かりやすい。
Androidではもっぱら非同期処理に使われている印象が強い。

RxKotlin

https://github.com/ReactiveX/RxKotlin
RxJavaのKotlinExtentionsを使えるライブラリ。
KotlinでRx使うなら導入すべき。

RxAndroid

https://qiita.com/oxsoft/items/9ae07c5512449b15b923
RxJavaをAndroidで使用するためのライブラリ。
使用例：https://qiita.com/oxsoft/items/9ae07c5512449b15b923

5. View関連

Groupie

https://github.com/lisawray/groupie
RecyclerViewでのアダプター生成時の手間を軽減できるライブラリ。
BindableItemを継承したクラスを、RecyclerViewのセルに該当するレイアウトに対してに使う。
使用例：https://qiita.com/orimomo/items/053524039cc63c3017a9

6. アーキテクチャ

MVVM

AndroidArchtectureComponentsでも使用しているアーキテクチャ。
Model,View,ViewModelに該当するフォルダ構成で、大雑把に言えばデータソース管理、
データ加工、データ表示で分類できる。

[View]
　　Activity, Fragment, Adapterなど
[Model]
　　WebAPI, LocalDataなど
[ViewModel]
　　データソースから取得したデータをビジネスモデルに変換する。
　　Viewレイヤーで使いやすく加工するイメージ。

参考URL：https://qiita.com/Tsutou/items/69a28ebbd69b69e51703

7. Jetpackの利用

DataBinding

https://developer.android.com/topic/libraries/data-binding?hl=ja
<layout>タグで囲むと、それらのレイアウトがDatabindingと認識され、
DataBindingImplクラスが生成される。
生成されたクラスから直接Viewにアクセスできて、findViewByIdなどしなくていよい。
結果、参照エラーなど凡ミスを防ぐことが期待できる。
<data>タグでViewModelや変数を定義することができ、FragmentやActivityから参照せずとも、
ViewModelでビジネスモデルの加工、表示が可能。

Roomについて

---

https://developer.android.com/training/data-storage/room?hl=ja
RoomはSQLiteを対象とする抽象レイヤーを提供しつつ、データベースのキャッシュを可能とします。
オフラインでRoomデータベースに変更が加えられた場合、オンライン時に自動的にWebと同期する仕組みがあるようです。
※福田はRoomを使った経験はないため断言できませんが、データベースのキャッシュと同期を割と簡単に実装できるかも。実験的に試してみてもよいか思います。

サンプルアプリ：https://github.com/android/sunflower
参考URL：https://tech.recruit-mp.co.jp/mobile/post-12311/