WebAPI 応用

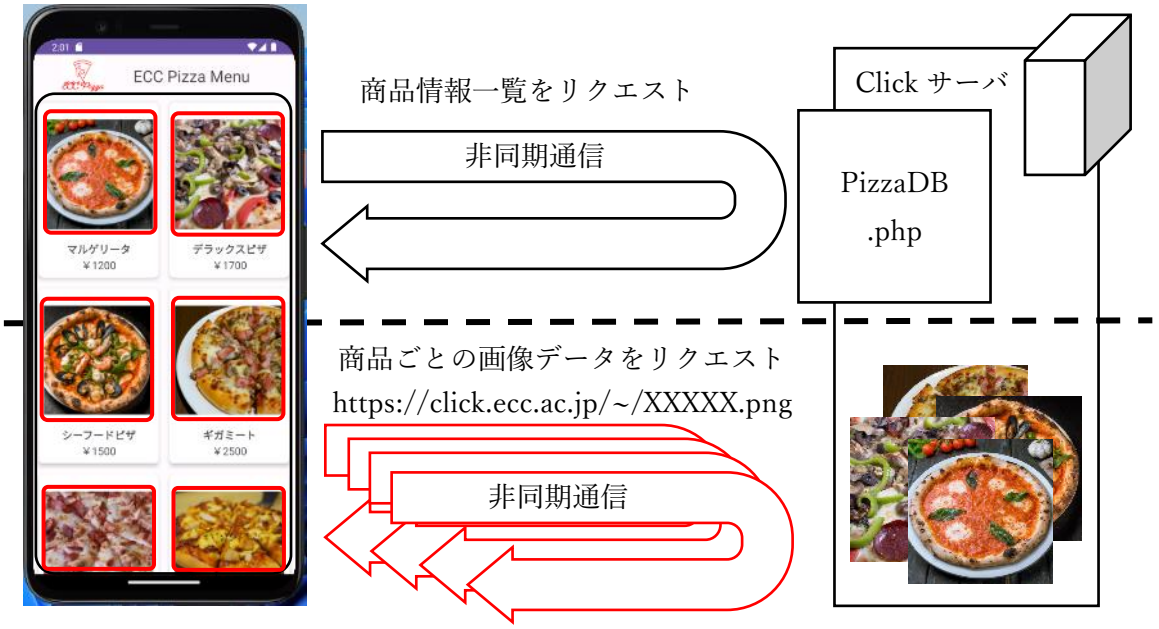
WebAPI に画像 URL が含まれている場合について

今回の中間課題では、Click サーバの WebAPI の取得内容からピザの商品情報を表示するアプリケーションを作成します。API レスポンスには例のように商品画像が含まれています。

・PizzaDB.php リクエスト取得情報(一部抜粋)

```
[
  {
    "product_no": "0001",
    "pname": "マルゲリータ",
    "category": "ピザ",
    "price": 1200,
    "image_url": "https://click.ecc.ac.jp/ecc/yishida/images/MargheritaPizza.png",
    "detail": "トマトの甘味とフレッシュバジルの香りが特徴。シンプルながらも贅沢な味わい。"
  },
  ～～～省略～～～
]
```

・画像取得イメージ



Web URL からの写真を表示するのは簡単に思えますが、適切に処理するにはかなりの作業が必要です。画像をダウンロードして内部的に保存(キャッシュ)して、Android が使用できる形式にデコードする必要があります。またこれらの処理をバックグラウンドのスレッドで行いつつ、UI の応答性を維持する必要もあります。

Android では、Glide や Coil などの画像処理ライブラリが利用可能なので、事情が無ければライブラリの導入を推奨します。

Glide の導入

Glide は、Android 用の高速かつ効率的なオープンソースのメディア管理および画像読み込みフレームワークです。メディアのデコード、メモリとディスクのキャッシュ、リソースプーリングをシンプルで使いやすいインタフェースで開発を行うことが出来ます。

・Glide 公式サイト
https://bumptech.github.io/glide/

Glide は、ビデオ静止画、画像、アニメーション GIF のフェッチ、デコード、表示をサポートしています。また、リモート画像の取得、サイズ変更、表示が必要なほぼすべての場合にも効果的です。Glide には、開発者がほぼすべてのネットワークスタックにプラグインできる柔軟な API が含まれています。

ハンズオン　プロジェクトに Glide を導入する

---

1. HandsOnWebAPI のプロジェクトを開く
2. App レベルの Gradle ファイルに Glide ライブラリを追加する

```
implementation("com.github.bumptech.glide:glide:4.16.0")
```

Glide の基本的な使い方

Glide を使用した画像の読み込みは簡単で、多くの場合で必要なのは 1 行だけです。

```
Glide.with(fragment).load(myUrl).into(imageView)
```

不要になったロードをキャンセルするのも簡単です。

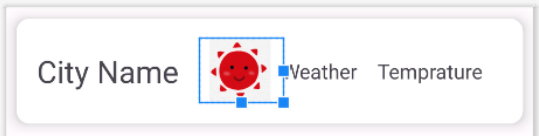
```
Glide.with(fragment).clear(imageView)
```

不要になったロードを削除することをお勧めしますが、そうする必要はありません。Glide は、渡されたアクティビティまたはフラグメントが破棄されると、自動的にロードをクリアし、ロードによって使用されたリソースをリサイクルします。

ListView または RecyclerView に画像をロードすると、単一の View にロードする場合と同じロード ラインが使用されます。Glide はビューの再利用とキャンセルのリクエストを自動的に処理します。

ハンズオン　Glide を使った画像リソースの取得

1．weather_item.xml に ImageView を追加する



コンポーネント	属性	設定値
ImageView	Id	weatherIcon
	Layout_width	50dp
	Layout_height	50dp
	srcCompat	@drawable/app_icon

2．ViewHolder に ImageView のオブジェクト変数を宣言する

```
val weatherIcon : ImageView
```

3．Init に初期化処理を追記する

```
weatherIcon = item.findViewById(R.id.weatherIcon)
```

【WebAPI 課題　未完成】

4．WeatherAdapter のクラス変数にアイコン情報リストを追加する

```
val icons =
listOf("01d","02d","03d","04d","09d","10d","11d","13d","50d","01n","02n")
```

5．onBindViewHolder メソッドでアイコン取得 URL を生成する

```
val iconURL =
"https://openweathermap.org/img/wn/${icons[position]}@2x.png"
```

6．Glide ライブラリを使用して、画像リソースを ImageView に適用させる

```
Glide.with(holder.itemView).load(iconURL).into(holder.weatherIcon)
```

【WebAPI 課題　完了】

4．onBindViewHolder メソッドでアイコン取得 URL を生成する

```
val iconURL =
"https://openweathermap.org/img/wn/${weatherList[position].weather[0].icon}@2x.png"
```

5．Glide ライブラリを使用して、画像リソースを ImageView に適用させる

```
Glide.with(holder.itemView).load(iconURL).into(holder.weatherIcon)
```