本日の内容

- RecyclerView
- Adapter
- (WebView)

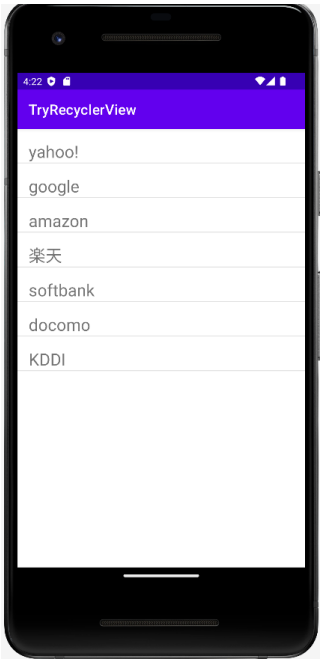
資料の環境

Android studio Dolphin

CompileSdkVersion:33

MinSdkVersion:23

今回はリストの使用方法と(ex にて Web ページの表示)を学んでいきます。

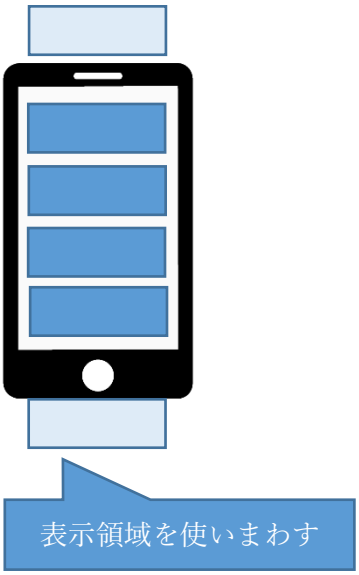


■RecyclerView とアダプター

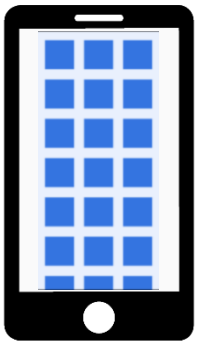
公式ドキュメント

https://developer.android.com/codelabs/kotlin-android-training-recyclerview-fundamentals?hl=ja#0

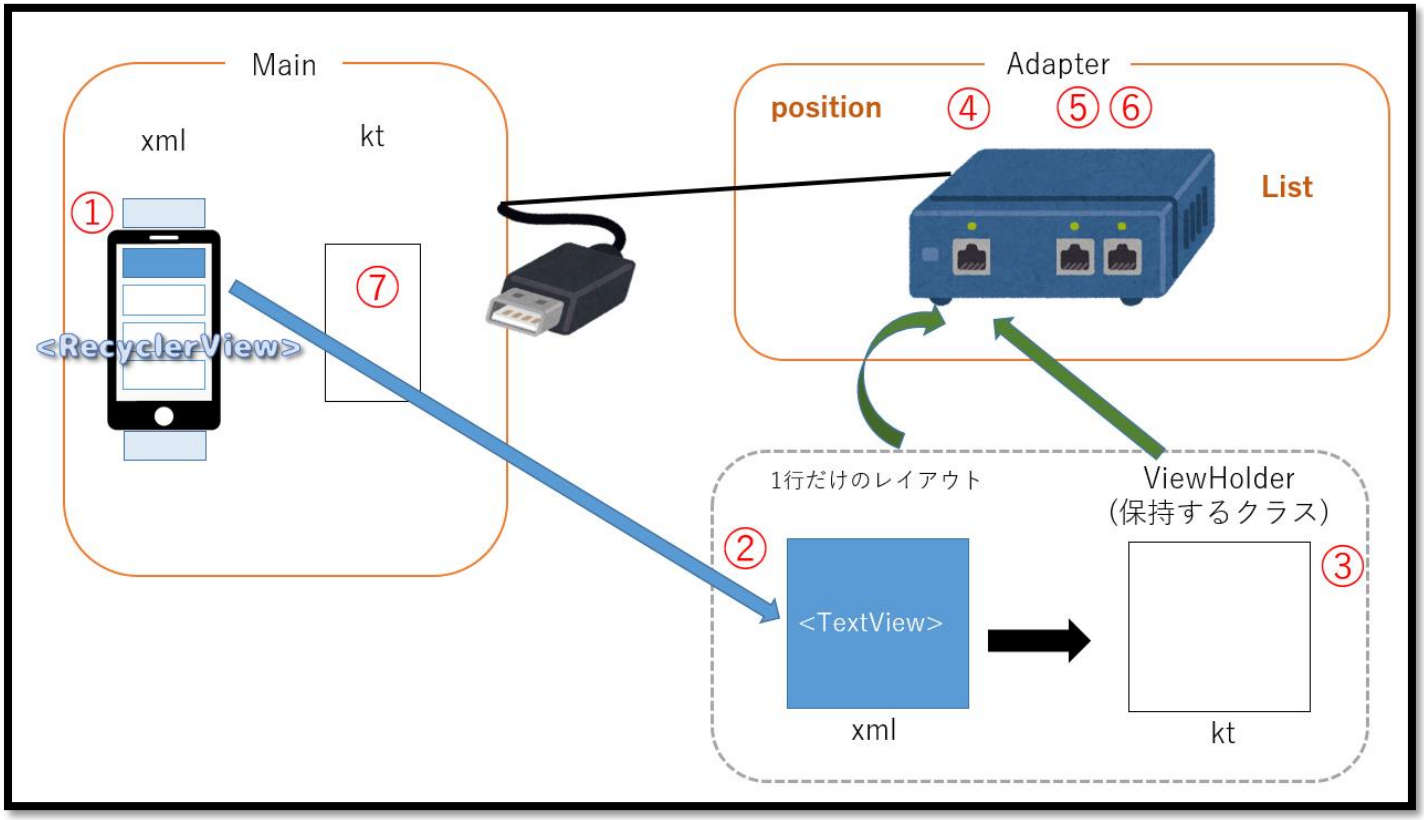
RecyclerView とは
データを表示する為のスクロール可能な View
RecyclerView.Adapter とは
View の作成、表示される View とデータの結び付けを行う
RecyclerView.ViewHolder とは
View への参照を保持 メンバ変数に View を持つ findViewById を毎回行う必要がない
RecycerView.LayoutManager とは
RecyclerView の中で View の位置や大きさの決定を行う



他にもこのようなリストも作成可能



◆各要素の概念



①Main の xml に RecyclerView のリストを作成
②1 行だけのレイアウトを作成
③レイアウト情報(View)を保持、管理するクラスを作成
④Adapter に各部品(View)をまとめる
⑤Positon 番目のデータをレイアウト(xml)に表示するようセット
⑥データが何件あるかカウント
⑦アダプターと接続しリストを表示

■Project 作成

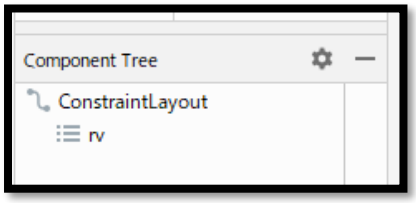
プロジェクト名：TryRecyclerView

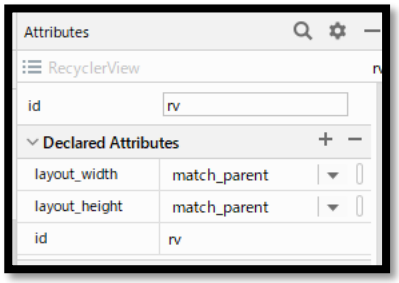
==========

◆① ：activity_main.xml

- デフォルトの textView を削除
- RecyclerView の配置

id:rv
layout_width:match_parent
layout_height:match_parent





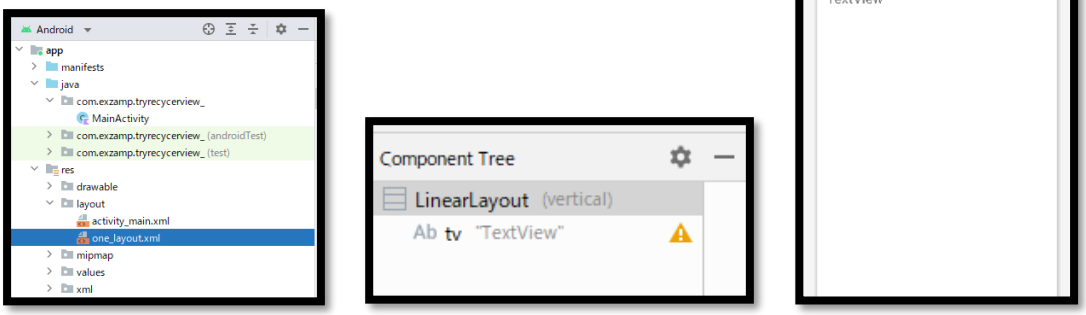
◆② : 1 行だけのレイアウト(xml)を作成

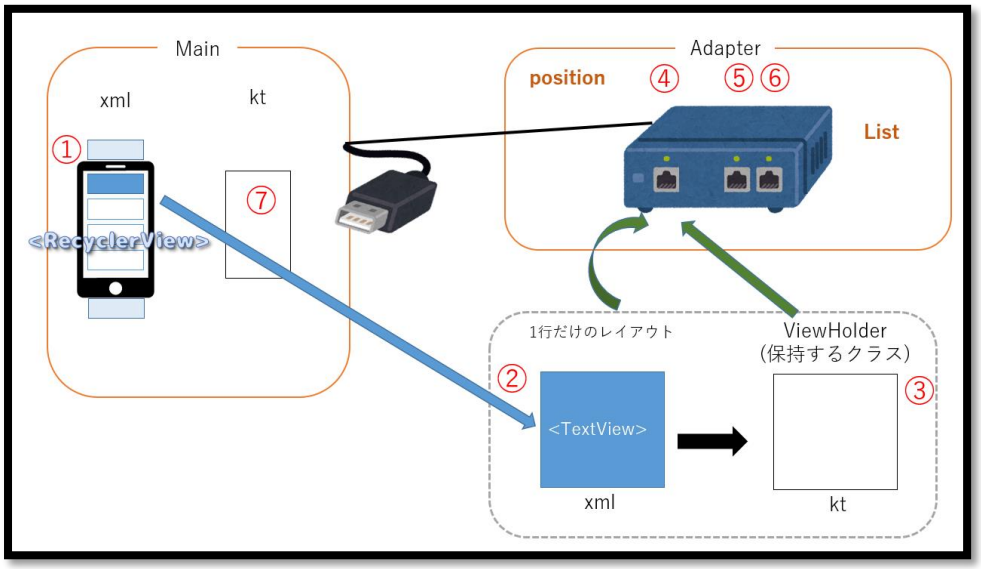
それでは、1 行だけのレイアウトを作成していきます。

レイアウトは res>layout を右クリック>new>layout Resourse File を選択

ファイル名：one_layout

- TextView の配置
  - id:tv
  - layout_width: match_parent
  - layout_height: wrap_content
  - layout_marginStart:16dp
  - layout_marginTop:16dp
  - textSize:24sp
- ルートタグの編集
- ルートタグを LinearLayout に変更
  layout_height: wrap_content に変更
  orientation を vertical に変更





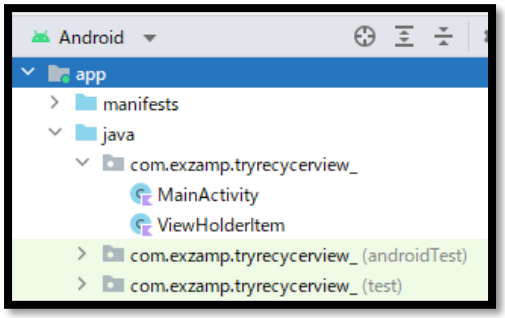
◆③： ViewHolder（レイアウト情報(View)を保持、管理する）クラスを作成

次に ViewHolder クラスを作成していきましょう！

MainActivity と同じパッケージを右クリック>new>java class を選択

ファイル名: ViewHolder

これで java クラスが出来ました。




```
package com.exzamp.tryrecyclerview

public class ViewHolder {
}
```

しかしこのままでは、RecyclerView の ViewHolder か判別できません

その為に RecyclerView を継承させるため、extends を追記し

続けて, RecyclerView の ViewHolder を継承したいので

RecyclerView.ViewHolder　と追記します

```
import androidx.recyclerview.widget.RecyclerView;

public class ViewHolder extends RecyclerView.ViewHolder{
}
```

これで継承は出来ました、super クラスのコンストラクタには引数が必要です。
引数ありのコンストラクタを作成しますが、Alt+Enter キーを押下して
自動で実装してしましましょう。

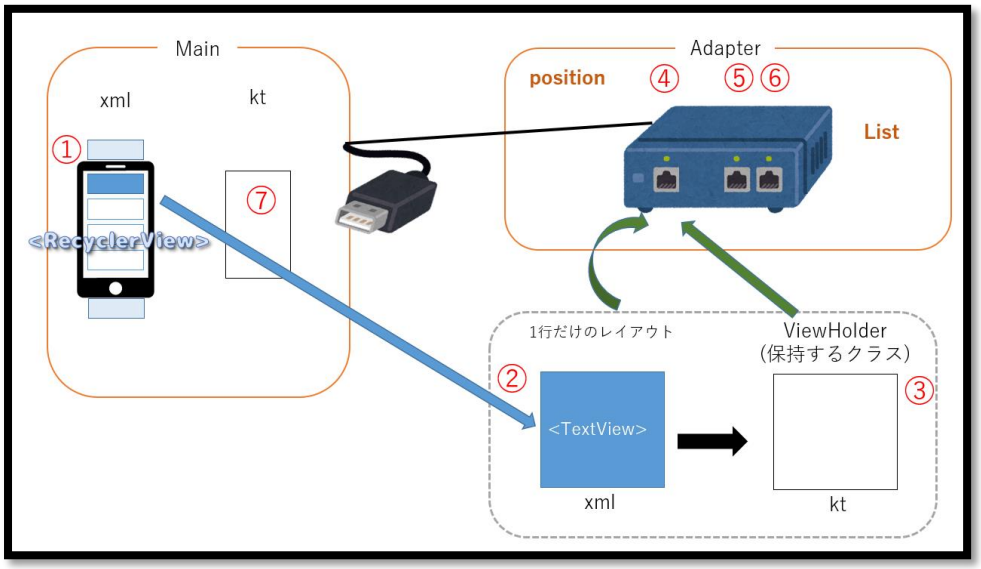
```
public class ViewHolder extends RecyclerView.ViewHolder{
    // コンストラクタ
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
    }
}
```

準備が出来ましたので、処理を記入していきます。
この ViewHolder.kt クラス内で
先ほど one_layout.xml で作った TextView の id を取得すればＯＫです。

今回 findViewById を直接使用することは出来ませんでした。
これは Activty を継承していない為、そのまま使えません。
しかし、引数で渡されている View を元に関連付けは出来るので
itemView.finadViewById としています。

```
public class ViewHolder extends RecyclerView.ViewHolder{
    // 宣言
    final TextView itemName;
    // コンストラクタ
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        // 関連付け
        this.itemName = itemView.findViewById(R.id.tv);
    }
}
```

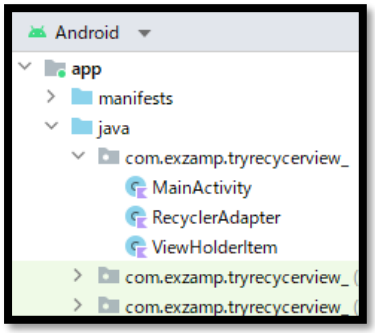
これで 1 行分のレイアウトは完成しました！
あとは Main の処理とこの Holder をつなぐアダプターというものを作成します。



◆④：Adapter(Adapter に各部品(View)をまとめる)

1 行だけのレイアウトの xml とそれを保持する Holder クラスを用意出来ましたので
それらをアダプターに繋げるという作業を行います。

アダプターのクラスを作成するので
MainActivity と同じパッケージを右クリック>new>java class を選択
ファイル名：RecyclerAdapter




```
package com.exzamp.tryrecyclerview

public class RecyclerAdapter {
}
```

これで java クラスが出来ました。
今回は RecyclerView パッケージ内の Adapter クラスを継承させます。
クラス名に続けて
extends RecyclerView.Adapter<ViewHolder>

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder>{
}
```

しかし、クラス名に赤い波線が出ています。
これは必要なメンバが足りていない為です。
赤波をクリックし[Alt]キーと[Enter]キーを押下すると「imprement members」が出てきますので選択。
それぞれ指示に従い 3 つ追加しましょう。コメント部分は不要なので削除してください

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder>{
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }
}
```

各メソッドの説明

onCreateViewHolder()
1 行分のレイアウト(View)を生成 ②と③を関連付ける
onBindViewHolder()
Position 番目のデータをレイアウト(xml)に表示するようセット
getItemCount()
データが何件あるかをカウント

◆④続き：アダプターの処理 その 1 : 1 行分のレイアウトを生成

④-1 : 1 行分のレイアウトを

LayoutInflater.from(元のコンテキスト)

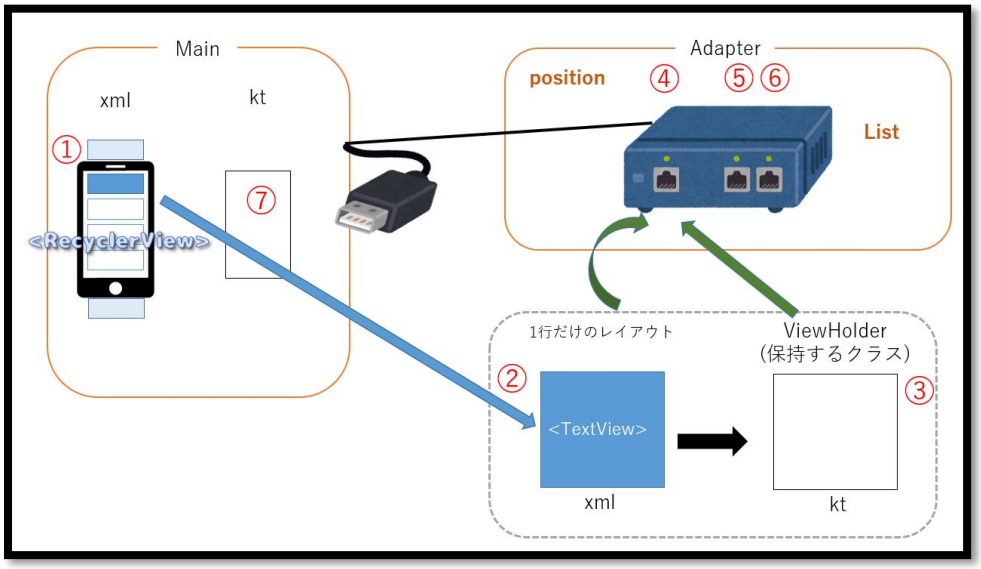
.inflate(1 行分のレイアウトの ID , ViewRoot , View グループに自動で set するか)

で取得

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder>{
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // 1行分のレイアウトを取得
        View itemXml = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.one_layout,parent, attachToRoot: false);
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

    }
```

④-2：戻り値として、今作成した ViewHolder クラスで返す

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder>{
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // 1行分のレイアウトを取得
        View itemXml = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.one_layout,parent, attachToRoot: false);
        return new ViewHolder(itemXml);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

    }
```



◆⑤：アダプターの処理 その２：positon 番目のデータをレイアウト(xml)に表示するようセット

⑤-1：アダプタークラス内に、表示するリストを定義

"yahoo!","google","amazon","楽天","softbank",

"docomo","KDDI"

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder> {

    // 表示するリストを作成
    static List<String> bookmarkList = new ArrayList<String>(Arrays.asList(
            "yahoo!","google","amazon","楽天","softbank",
            "docomo","KDDI"
    ));

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // 1行分のレイアウトを取得
        View itemXml = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.one_layout, parent, attachToRoot: false);
        return new ViewHolder(itemXml);
    }
```

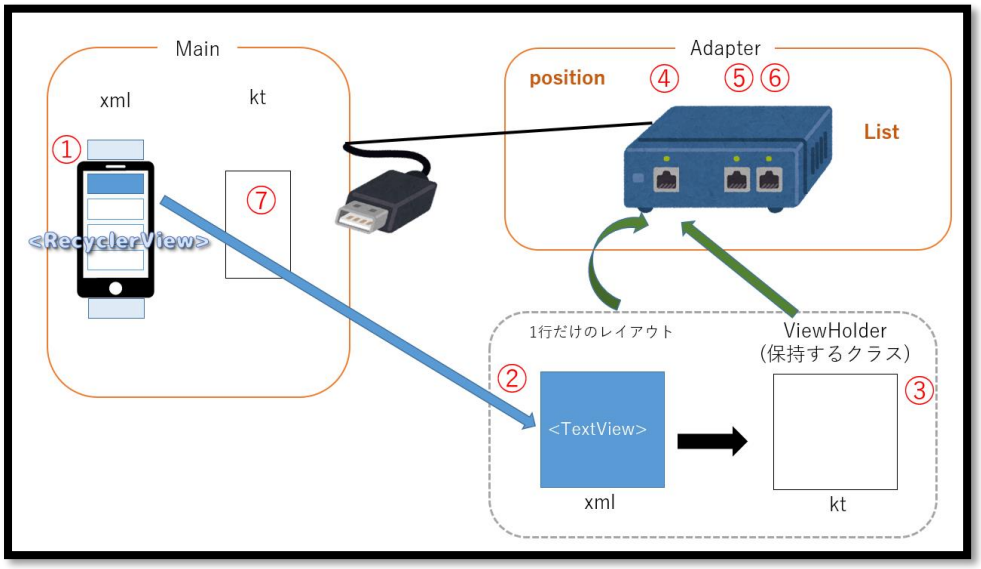
⑤-2：RecyclerAdapter クラスの onBindViewHolder メソッド内に

holder.itemName.setText＝ 定義したリスト.get(positon)

と設計することにより

positon 番目の item テキストにリスト[position 番目]のテキストをセット

```
// position番目のデータをレイアウト(xml)に表示するようセット
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    // position番目のitemテキストにリスト[position番目]のテキストをセット
    holder.itemName.setText(bookmarkList.get(position));
}
```

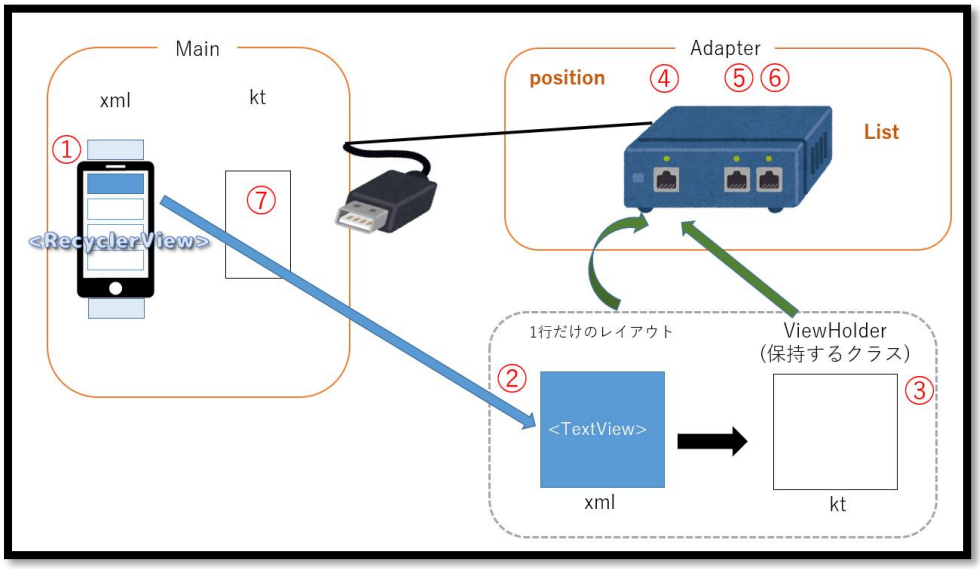


3つ目のメソッドも処理を書き換えていきましょう。

◆⑥：アダプターの処理 その3：データが何件あるかカウント

⑥-1：　戻り値に定義したリストの要素数を
リスト.size
と使用することで要素数を返すことが出来ます。

```
// データ数が何件あるか返すメソッド
@Override
public int getItemCount() {
    //  リストのデータ数を返す
    return bookmarkList.size();
}
```



◆⑦リストの表示

⑦-1 : MainActivity クラス内に RecyclerView の変数を用意

```
public class MainActivity extends AppCompatActivity {
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

⑦-2 : xml の RecyclerView と関連付け

⑦-3 : RecyclerView の Adapter に作成した Adapter をセット

```
public class MainActivity extends AppCompatActivity {
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // xmlのRecyclerViewと関連付け
        recyclerView = findViewById(R.id.rv);
        // RecyclerViewのAdapterに作成した自作Adapterをセット
        recyclerView.setAdapter(new RecyclerAdapter());
    }
}
```

⑦-4：アイテムの並べ方をセット

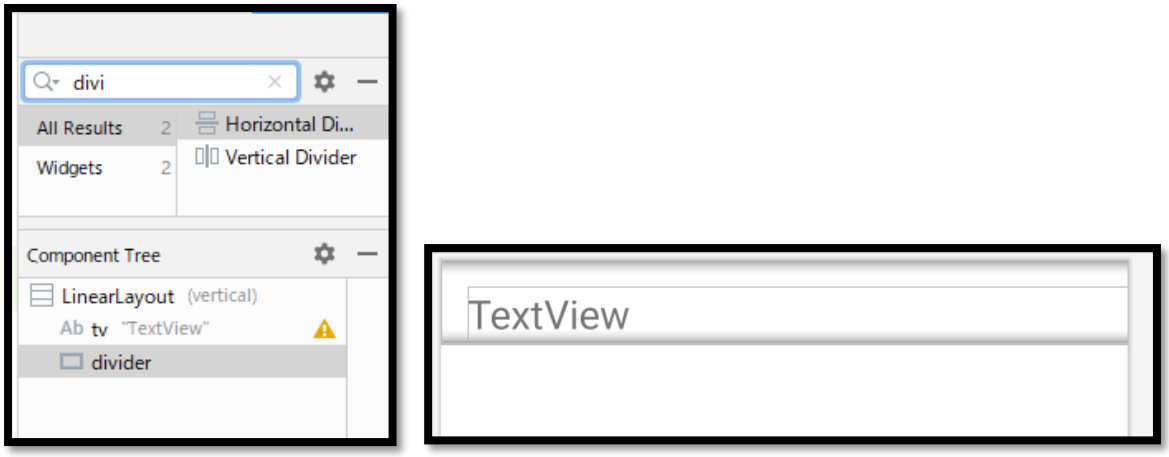
縦に並べる：LinearLayoutManager

パネルのように並べる：GridLayoutManager

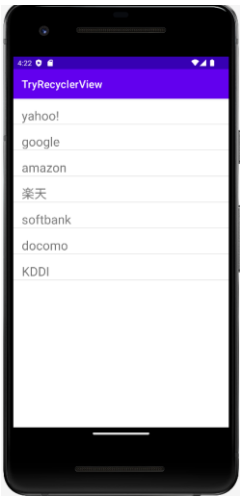
```
public class MainActivity extends AppCompatActivity {
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // xmlのRecyclerViewと関連付け
        recyclerView = findViewById(R.id.rv);
        // RecyclerViewのAdapterに作成した自作Adapterをセット
        recyclerView.setAdapter(new RecyclerAdapter());
        // アイテムの並べ方をセット
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    }
}
```

◆⑧区切り線

⑧-1 : 1 行分のレイアウトに Horizontal Divider(区切り線)を CompornetTree 経由で配置



再度実行すると item 毎に線が引かれるようになりました！



◆⑨クリック処理(1 行分の画面(view)が押されたら)

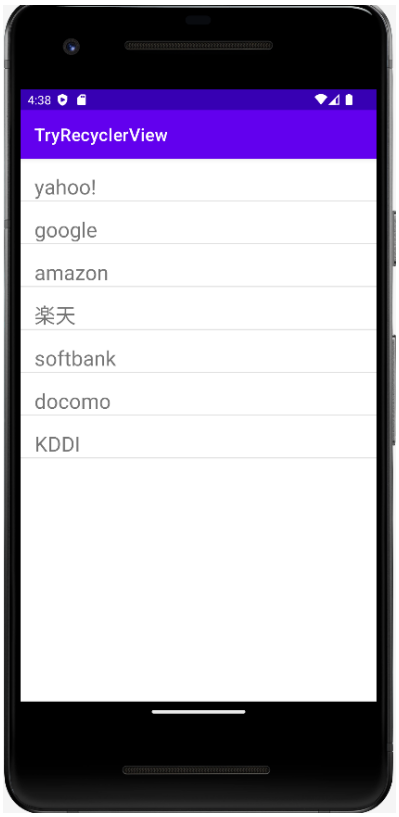
⑨-1 : itemView を押された時の処理を定義

```
public class ViewHolder extends RecyclerView.ViewHolder{
    // 宣言
    final TextView itemName;
    WebView webView;
    // コンストラクタ
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        // 関連付け
        this.itemName = itemView.findViewById(R.id.tv);
        // itemViewを押された時の処理
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });
    }
}
```

⑨-4 : 押された番号を取得

⑨-5 : 押されたテキストを Toast で表示

```
// コンストラクタ
public ViewHolder(@NonNull View itemView) {
    super(itemView);
    // 関連付け
    this.itemName = itemView.findViewById(R.id.tv);
    // itemViewを押された時の処理
    itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // 押された番号を取得
            int position = getAdapterPosition();
            // 押されたテキストをToastで表示
            Toast.makeText(view.getContext(),
                    RecyclerAdapter.bookmarkList.get(position),
                    Toast.LENGTH_SHORT)
                    .show();
        }
    });
}
```



リストが表示され Toast も表示されるようになりました！

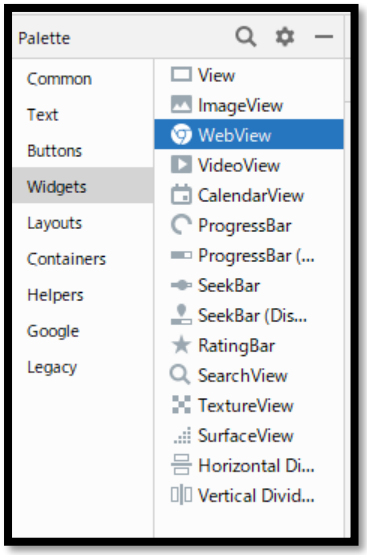
ここから ex

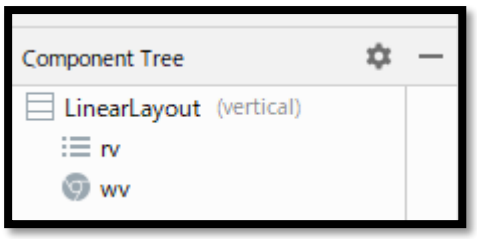
このままでは、ブックマークと言っているのに全然 web ページを使っていませんね。
実際に web ページを表示させ、ブックマークと同じような動きに追加修正を行っていきましょう！

■Web ページを表示する「WebView」
以前 Web アプリを起動するアプリを作成しましたが、あれは別アプリを起動しているだけで
作成しているアプリ内では Web の表示を行っていません。

アプリ内で表示するには
・WebView のウィジェットを使用する
・プログラム内で表示する URL の指定
・AndroidManifest にインターネット通信を行う権限を付与する
上記の 3 つが必要となります。

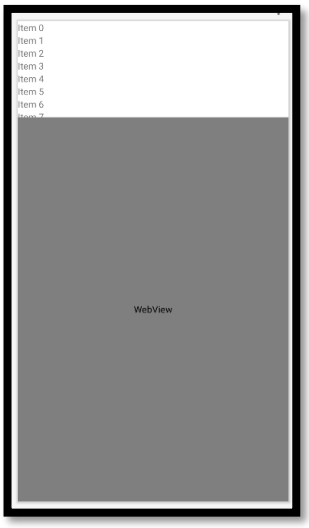
まずは WebView の配置
ルートタグを LinearLayout(vertical)に変更





WebView
　　id を wv に変更
　　layout_height を 0dp に変更(割合にする為)
　　layout_weight を 0.8

RecyclerView(rv)
　　layout_height を 0dp に変更(割合にする為)
　　layout_weight を 0.2



■表示する URL の指定

次に Activity 内で WebView の操作を行っていきましょう

変数の定義と関連付けを行います

```
public class MainActivity extends AppCompatActivity {
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;
    // WebViewの変数を定義
    private WebView webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // xmlのRecyclerViewと関連付け
        recyclerView = findViewById(R.id.rv);
        // RecyclerViewのAdapterに作成した自作Adapterをセット
        recyclerView.setAdapter(new RecyclerAdapter());
        // アイテムの並べ方をセット
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
        // 関連付け
        webView = findViewById(R.id.wv);
    }
}
```

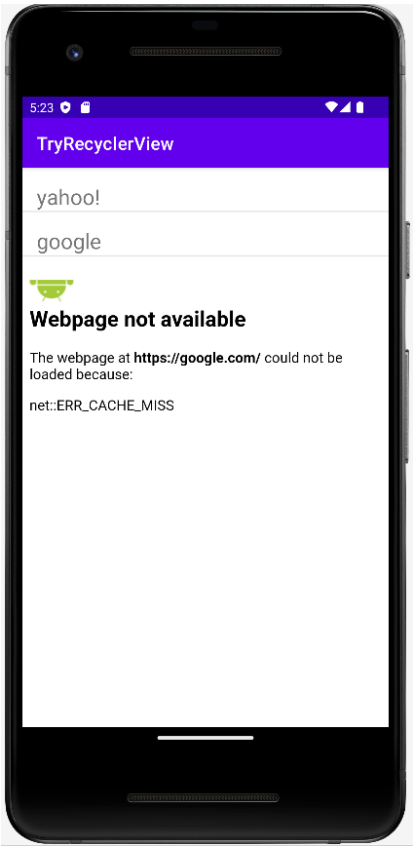
表示する URL も決めておきましょう

```
public class MainActivity extends AppCompatActivity {
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;
    // WebViewの変数を定義
    private WebView webView;
    // 初期URLを指定
    private String url = "https://google.com";
```

次に WebView の設定を付与していきます。

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // xmlのRecyclerViewと関連付け
    recyclerView = findViewById(R.id.rv);
    // RecyclerViewのAdapterに作成した自作Adapterをセット
    recyclerView.setAdapter(new RecyclerAdapter());
    // アイテムの並べ方をセット
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    // 関連付け
    webView = findViewById(R.id.wv);
    // WebViewを設定する
    webView.setWebViewClient(new WebViewClient());
    // JavaScriptを有効にする
    webView.getSettings().setJavaScriptEnabled(true);
    // urlを読み込む
    webView.loadUrl(url);

}
```

これで実行出来ると思いきや、エラーが発生します。



エラーが出たので先生助けて！とならないようにエラーを元に Google などで調べてみましょう
すると権限がないことに辿り着きます。

■インターネット通信の権限付与
アプリに権限を付与するには AndroidManifest に記述が必要です。

<uses-permission android:name="android.permission.INTERNET"/>

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="TryRecyclerView"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TryRecyclerView"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
```

再度実行してみると、

指定した URL が表示されるようになりました！
しかし、yahoo!などを押しても webpage は移動しませんので、まだブックマークとしてはダメですね。

■多次元の ArrayList で管理
名前だけでなく、URL も一緒に管理してしまいましょう。

```
public class RecyclerAdapter extends RecyclerView.Adapter<ViewHolder> {

    // 表示するリストを作成
//    List<String> bookmarkList = new ArrayList<String>(Arrays.asList(
//            "yahoo!","google","amazon","楽天","softbank",
//            "docomo","KDDI"
//    ));
    // 表示するURLと表示名のリストMapを作成
    static List<ArrayList<String>> bookmarkList = new ArrayList<ArrayList<String>>(
            Arrays.asList(new ArrayList<String>(Arrays.asList("yahoo!", "https://www.yahoo.co.jp/")),
                    new ArrayList<String>(Arrays.asList("google", "https://www.google.com/")),
                    new ArrayList<String>(Arrays.asList("amazon", "https://www.amazon.co.jp/")),
                    new ArrayList<String>(Arrays.asList("楽天", "https://www.rakuten.co.jp/")),
                    new ArrayList<String>(Arrays.asList("softbank", "https://www.softbank.jp/")),
                    new ArrayList<String>(Arrays.asList("docomo", "https://www.docomo.ne.jp/")),
                    new ArrayList<String>(Arrays.asList("KDDI", "https://www.kddi.com/"))
            )
    );
```

2 次元配列になったので、リスト表示用の 0 番目の値をセットに変更しましょう

RecyclerAdapter.java

```java
// position番目のデータをレイアウト(xml)に表示するようセット
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    // position番目のitemテキストにリスト[position番目]のテキストをセット
    holder.itemName.setText(bookmarkList.get(position).get(0));
}
```

ViewHolder.java

```java
public class ViewHolder extends RecyclerView.ViewHolder{
    // 宣言
    final TextView itemName;
    WebView webView;
    // コンストラクタ
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        // 関連付け
        this.itemName = itemView.findViewById(R.id.tv);
        // itemViewを押された時の処理
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // 押された番号を取得
                int position = getAdapterPosition();
                // 押されたテキストをToastで表示
                Toast.makeText(view.getContext(),
                        RecyclerAdapter.bookmarkList.get(position).get(0),
                        Toast.LENGTH_SHORT)
                        .show();
            }
```

■Holder クラスから Activity 内のメソッドを実行
リスト内の処理は adapter もしくは holder 内のデータを使用して行います。
しかし、WebView があるのは MainActivity クラス、どうにかして Holder クラスから
Acrtivity クラスのメソッドを実行する必要があります。
その方法の 1 つとして今回はインターフェースを経由しメソッドを実行します。

■インターフェースクラスの生成
New>Javaclass>Interface
ファイル名を ICallWebView で作成しましょう。
更に抽象メソッドの setWebViewURL メソッドを定義し、引数として Stinrg 型を 1 つ渡すように設定します

```
public interface ICallWebView {
    // WebViewへuelをセットする抽象メソッド
    void setWebViewURL(String url);
}
```

■Activity クラスにインターフェースを実装
先程の ICallWebView インターフェースを MainActivity クラスに実装していきましょう。

```
public class MainActivity extends AppCompatActivity implements ICallWebView{
    // RecyclerViewの変数を定義
    private RecyclerView recyclerView;
    // WebViewの変数を定義
    private WebView webView;
    // 初期URLを指定
    private String url = "https://google.com";
```

まだ、先ほど作成した抽象メソッド setWebViewURL の実装がまだですので実装しましょう

MainActivity.java

```
        // 関連付け
        webView = findViewById(R.id.wv);
        // WebViewを設定する
        webView.setWebViewClient(new WebViewClient());
        // JavaScriptを有効にする
        webView.getSettings().setJavaScriptEnabled(true);
        // urlを読み込む
        webView.loadUrl(url);
    }

    // WebViewへuelをセットするメソッド
    @Override
    public void setWebViewURL(String url) {

    }
}
```

渡された String 型の URL で webView のアクセス先を変更してみましょう

```
        // 関連付け
        webView = findViewById(R.id.wv);
        // WebViewを設定する
        webView.setWebViewClient(new WebViewClient());
        // JavaScriptを有効にする
        webView.getSettings().setJavaScriptEnabled(true);
        // urlを読み込む
        webView.loadUrl(url);
    }

    // WebViewへuelをセットするメソッド
    @Override
    public void setWebViewURL(String url) {
        // urlを読み込む
        webView.loadUrl(url);
    }
}
```

この状態で実行しても、特に変化がありません。

インターフェースを実装しましたが、
どのクラスからインターフェース経由でメソッドを呼び出す処理が現状はありません。

■リスナーの宣言
どのリスナーなのか判別させるために宣言を行います。

```
public class ViewHolder extends RecyclerView.View
    // 宣言
    final TextView itemName;
    // インターフェースの宣言
    static ICallWebView listener;
    // コンストラクタ
    public ViewHolder(@NonNull View itemView) {
```

■リスナーの登録

MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // xmlのRecyclerViewと関連付け
    recyclerView = findViewById(R.id.rv);
    // RecyclerViewのAdapterに作成した自作Adapterをセット
    recyclerView.setAdapter(new RecyclerAdapter());
    // アイテムの並べ方をセット
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    // 関連付け
    webView = findViewById(R.id.wv);
    // WebViewを設定する
    webView.setWebViewClient(new WebViewClient());
    // JavaScriptを有効にする
    webView.getSettings().setJavaScriptEnabled(true);
    // urlを読み込む
    webView.loadUrl(url);
    // リスナーの登録
    ViewHolder.listener = this;
}
```

リスナーの登録をしたのに何も変わりません。

何故なら、リスナー登録はしましたが実際に setWebViewURL を呼び出す処理を書いていないからです。

■setWebViewURL メソッドの呼び出し
実際に必要なのは何番目のリストが押されたかの処理がある
ViewHolder.java クラス内の onClick メソッド内で実行する必要があります。

```
        this.itemName = itemView.findViewById(R.id.tv);
        // itemViewを押された時の処理
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // 押された番号を取得
                int position = getAdapterPosition();
                // 押されたテキストをToastで表示
                Toast.makeText(view.getContext(),
                        RecyclerAdapter.bookmarkList.get(position).get(0),
                        Toast.LENGTH_SHORT)
                        .show();
                // インターフェース経由でMainActivity内のsetWebViewURLメソッドを呼び出し
                // 押されたリストのURLをWebViewでアクセスする
                listener.setWebViewURL(RecyclerAdapter.bookmarkList.get(position).get(1));
            }
        });
    }
}
```

押下したボタンのサイトを表示するようになりました！