8.2 オプションメニュー

MenuSampleの基本部分ができたので、本章のテーマの1つであるオプションメニューをここから追加していきます。

8.2.1 オプションメニューの例

オプションメニューとは、**アクションバー**に表示されるメニューのことです。図8.5のように、画面上部のバー部分を**アクションバー**と呼び、そこにメニューを表示することが可能です。

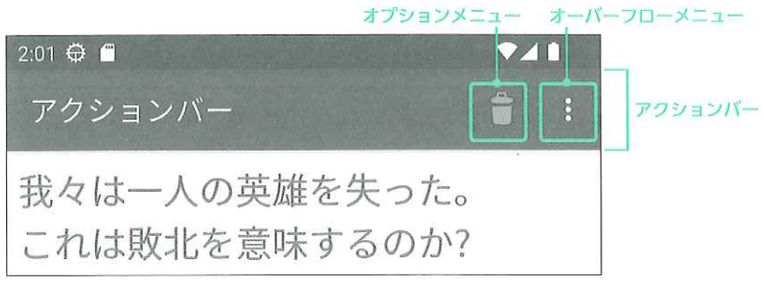

図8.5 アクションバーとオプションメニューの例

図8.5ではゴミ箱アイコンのメニューと右端の ⋮ アイコンのメニューが表示されています。この ⋮ アイコンのメニューのことを**オーバーフローメニュー**と呼び、これをタップすることでさらに選択肢が表示される仕組みとなっています。

今回のサンプルでは、このオーバーフローメニューをタップすると、図8.6のように定食とカレーを選択できるようになっており、それぞれを選択すると、選択されたメニューリストが表示されるように改造していきます。

なお、オプションメニューそのものは、画面と同じように.xmlファイルに記述します。

図8.6 今回のサンプルでオプションメニューが追加されたリスト画面

8.2.2 手順 オプションメニュー表示を実装する

1 menuファイルを格納するフォルダを作成する

まず、menu用の.xmlファイルを入れるフォルダを追加します。
resフォルダを右クリックし、

[New] → [Android Resource Directory]

を選択してください。図8.7のようなダイアログが表示されるので、「Resource type:」から「menu」を選択し、[OK] をクリックします。

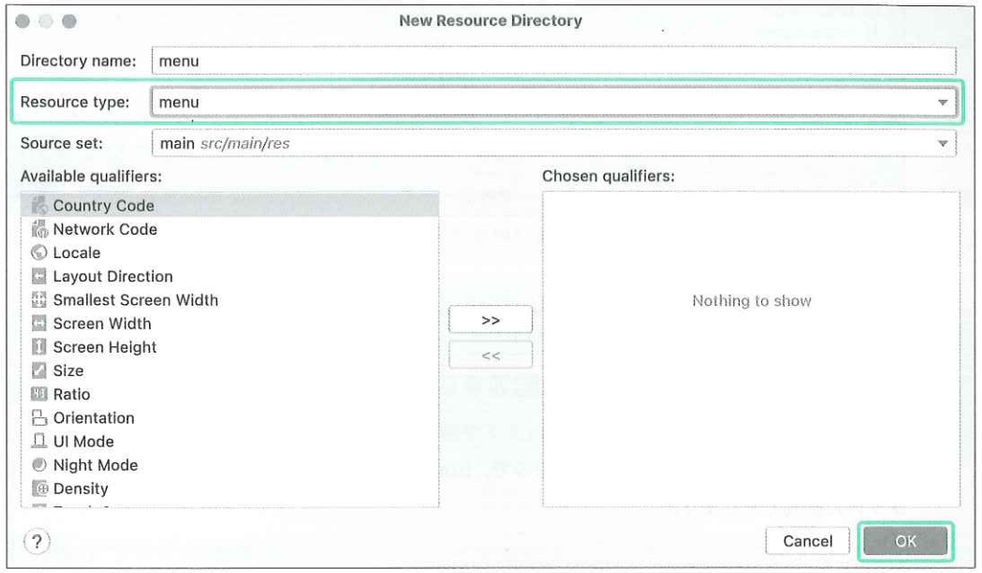

図8.7 リソースフォルダ追加画面

これでres/menuフォルダが追加されました。メニューに関係する.xmlファイルは、このフォルダ内に格納します。

2 menu用の.xmlファイルを作成する

次に、.xmlファイルを作成します。menuフォルダを右クリックし、

[New] → [Menu Resource File]

を選択してください。図8.8のようなダイアログが表示されるので、[File name:] に「menu_options_

menu_list」を入力し、[OK]をクリックします。

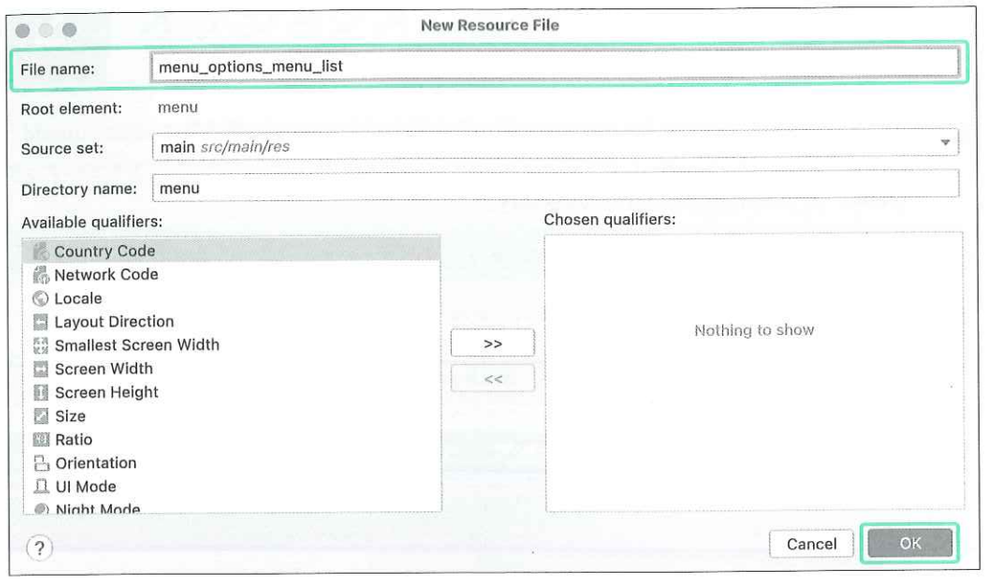

図8.8　メニュー用.xmlファイル追加画面

menuタグが記述された.xmlファイルが作成されます。

3 menu用のXMLタグを記述する

メニュー用の.xmlファイルは、menuタグで始まり、この中に選択肢1つにつきitemタグを1つ記述していきます。今回は、選択肢が2つなので、itemタグを2つ追加します。リスト8.5のコードをmenuタグ内に記述しましょう。

リスト8.5　res/menu/menu_options_menu_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/menuListOptionTeishoku"
        app:showAsAction="never"
        android:title="@string/menu_list_options_teishoku"/>
    <item
        android:id="@+id/menuListOptionCurry"
        app:showAsAction="never"
        android:title="@string/menu_list_options_curry"/>
</menu>
```


NOTE xmlns:app属性のインポート

menu開始タグのxmlns:app属性は、最初は記述されていません。itemタグのapp:showAsAction属性を記述する際にappが赤文字で表示され、図8.Aのようなメッセージが表示されます。

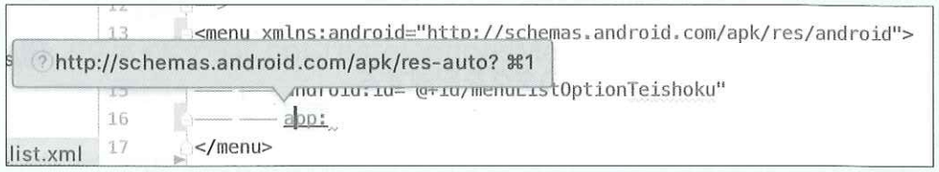

図8.A　appのエラー表示

そのときに、メッセージ通りにmacOSの場合は[⌘]+[1]キー、Windowsの場合は[Alt]+[Enter]キーを押すと、自動でインポートしてくれます。

4 アクティビティに記述する

メニュー用の.xmlファイルの記述ができたところで、今度は、アクティビティクラスにコードを記述します。MainActivityにリスト8.6のメソッドを追加しましょう。

リスト8.6　java/com.websarva.wings.android.menusample/MainActivity.kt

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    //オプションメニュー用xmlファイルをインフレート。
    menuInflater.inflate(R.menu.menu_options_menu_list, menu) ―― ❶
    return true ―― ❷
}
```

5 アプリを起動する

入力を終え、特に問題がなければ、この時点で一度アプリを実行してみてください。図8.5のようにメニューが表示されます。

8.2.3 オプションメニュー表示はXMLとアクティビティに記述する

オプションメニューを表示させるには、以下の手順を踏みます。

1. オプションメニュー用の.xmlファイルを作成する。
2. .xmlファイルに専用のタグを記述する。
3. アクティビティにonCreateOptionsMenu()メソッドを実装する。

❸［再生］ボタンの配置

Palette	Button	
Attributes	id	btPlay
	layout_width	0dp
	layout_height	wrap_content
	text	@string/bt_play_play
	enabled	false
	onClick	onPlayButtonClick
制約ハンドル	上	parent(8dp)
	左	btBackの右(8dp)
	右	btForwardの左(8dp)

❹「リピート再生」スイッチの配置

Palette	Switch	
Attributes	id	swLoop
	layout_width	wrap_content
	layout_height	wrap_content
	text	@string/sw_loop
制約ハンドル	上	btBackの下(8dp)
	左	parent(8dp)

その後、コードモードに切り替えて、❹で配置したSwitchタグを、リスト12.2の太字のように、SwitchMaterialタグへと変更してください。

リスト12.2 res/layout/activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
        :
    <com.google.android.material.switchmaterial.SwitchMaterial
        android:id="@+id/swLoop"
            :
</androidx.constraintlayout.widget.ConstraintLayout>
```

すべての画面部品を配置し終えると、レイアウトエディタ上では、図12.2のように表示されます。

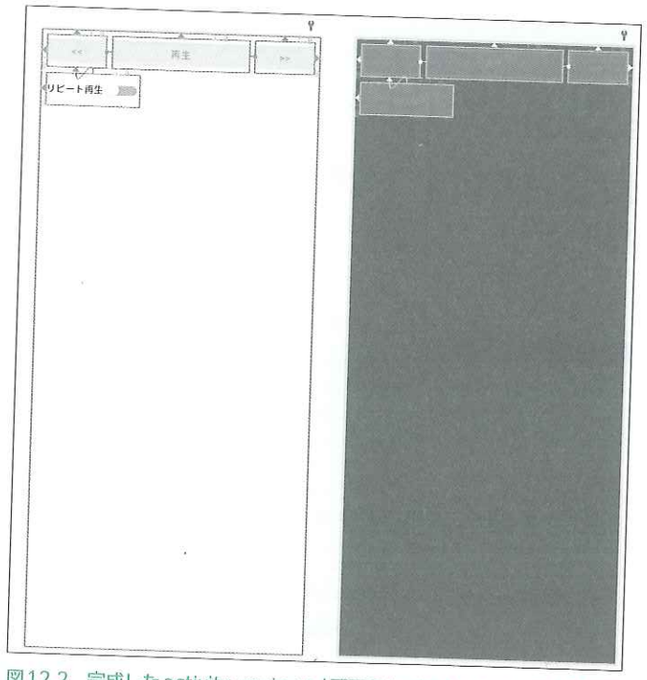

図12.2 完成したactivity_main.xml画面のレイアウトエディタ上の表示

4 音声ファイルを追加する

今回は音声ファイルを使います。以下の効果音フリー素材サイトから好きな音声ファイルをダウンロードしてください。本サンプルでは「渓流」を使います。

●効果音ラボ（環境音のページ）

https://soundeffect-lab.info/sound/environment/

ダウンロードした音声ファイルをMediaSampleプロジェクトのリソースファイルとして格納します。その際、Androidのコーディング規約として、リソースファイル名には小文字とアンダーバーのみしか使えません。そこで、適当にリネームします。ここでは、「mountain-stream1.mp3」というダウンロードファイル名を「mountain_stream.mp3」に変更しています。

リネームが済んだファイルを、resフォルダ配下にrawフォルダを作成し、このフォルダに格納します（図12.3）。rawフォルダを作成するには、リソースフォルダの追加画面で［Resource type:］から［raw］を選択します。

参照 リソースフォルダの追加 ➡ 8.2.2項 手順 1 p.197

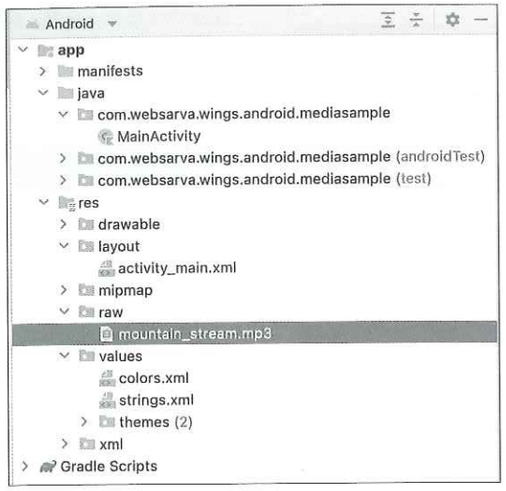

図12.3　音声ファイルを格納したプロジェクト構成

作成したフォルダにファイルを格納するには、ファイルシステム（Windowsならエクスプローラー、MacならFinder）上で音声ファイルをコピーし、Android Studioのプロジェクトツールウィンドウ上のrawフォルダを選択してペーストします。すると、Android Studioがコピー確認のダイアログを表示するので、特に問題がなければそのまま［OK］をクリックします。

5 アプリを起動する

入力を終え、特に問題がなければ、この時点で一度アプリを実行してみてください。図12.4の画面が表示されます。

図12.4　ここまでのコードで表示される画面

現段階では、ボタンはタップできないようになっています。これは、手順3で各Buttonを配置する際、そのenabled属性として、falseを指定しているからです。ここから、再生ボタンをタップすると、rawフォルダに格納した音声ファイルが再生されるようにソースコードを記述していきます。その際、音声ファイルの再生準備が整うまで、ボタンが押されないようにしてあるのです。

12.1.2 手順 メディア再生のコードを記述する

では、いよいよ音声ファイルを再生するコードを記述しましょう。

1 メディアプレーヤー準備のコードを記述する

MainActivityクラスに、リスト12.3のようにプロパティを追加し、onCreate()メソッド内にコードを追記しましょう※1。

リスト12.3　java/com.websarva.wings.android.mediasample/MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    //メディアプレーヤープロパティ。
    private var _player: MediaPlayer? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //プロパティのメディアプレーヤーオブジェクトを生成。
        _player = MediaPlayer()                                          ❶
        //音声ファイルのURI文字列を作成。
        val mediaFileUriStr = "android.resource://${packageName}/${R.raw.mountain_stream}"  ❷-1
        //音声ファイルのURI文字列を元にURIオブジェクトを生成。
        val mediaFileUri = Uri.parse(mediaFileUriStr)                    ❷-2
        // プロパティのプレーヤーがnullでなければ…
        _player?.let {                                                   Ⓐ
            // メディアプレーヤーに音声ファイルを指定。
            it.setDataSource(this@MainActivity, mediaFileUri)            ❷-3
            // 非同期でのメディア再生準備が完了した際のリスナを設定。
            it.setOnPreparedListener(PlayerPreparedListener())           ❸-1
            // メディア再生が終了した際のリスナを設定。
            it.setOnCompletionListener(PlayerCompletionListener())       ❸-2
            // 非同期でメディア再生を準備。
            it.prepareAsync()                                            ❹
        }
    }
}
```

2 リスナメンバクラスを追加する

手順1を記述した際、PlayerPreparedListenerクラスとPlayerCompletionListenerクラスがないためコンパイルエラーとなっています。これらのクラスを、メンバクラスとしてMainActivityクラスに追記しましょう（リスト12.4）。

※1　この時点ではまだPlayerPreparedListenerクラスとPlayerCompletionListenerクラスを作成していないため、コンパイルエラーになります。これらのクラスは次の手順で記述します。