

Android Tutorial using Kotlin 第六堂

(1) Material Design - Theme與Transition

[Android Tutorial using Kotlin 第五堂 \(3\) 設計小工具元件 – AppWidget << 前情](#)

2014年的Google開發人員大會，發表新一代的版本Android L，也就是Android 5 Lollipop的預覽版本。Android 5 Lollipop除了許多新的功能外，也同時發表Material Design，針對Android應用程式的畫面與操作，提供創新與統一的設計方式。應用程式員可以設計具有3D空間感覺，與使用元件與畫面的轉換動畫效果，讓使用者在操作應用程式的時候更加順暢，畫面也更加美觀。

這一章開始介紹從Android 5 Lollipop開始提供的Material Design，套用Material Design提供的樣式，可以簡化應用程式畫面的設計，也可以統一畫面的風格。下列是目前設計好的記事應用程式畫面：



18-1 Material Design Theme

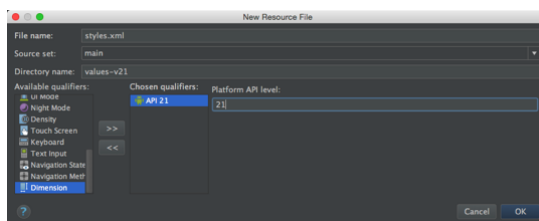
Android 5 Lollipop開始提供全新的Material Design樣式，讓應用程式可以直接套用下列三種樣式：

- **android:style/Theme.Material**：預設的樣式，暗色樣式

- **android:style/Theme.Material.Light**：亮色樣式
- **android:style/Theme.Material.Light.DarkActionBar**：亮色樣式搭配暗色的功能表

接下來直接把Material Design樣式套用在記事應用程式。上列說明的樣式在Android 5 Lollipop、API Level 21開始提供，為了的版本，依照下列的步驟，建立一個新版本使用的樣式資源：

1. 在「res/values」目錄按滑鼠右鍵 -> New -> Values Resource File。
2. 在「File Name:」欄位輸入「styles.xml」。
3. 在「Available qualifiers:」選擇「Version」。
4. 選擇「>>」按鈕。
5. 在「Platform API Level」欄位輸入「21」。
6. 選擇「OK」按鈕。



依照下列的內容，修改預設的樣式資源檔：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material">

        </style>
</resources>
```

目前Material Design還沒有支援ActionBar，所以要執行相關的修改。開啟「MainActivity.kt」，原來繼承自「AppCompatActivity」類別的部份，改為繼承自「Activity」

```
package net.macdidi.atk
...
class MainActivity : Activity() {
    ....
}
```

開啟「ItemActivity.kt」，原來繼承自「AppCompatActivity」類別的部份，改為繼承自「Activity」

```
package net.macdidi.atk

...
class ItemActivity : Activity() {
    ....
}
```

開啟「MapsActivity.kt」，原來繼承自「AppCompatActivity」類別的部份，改為繼承自「FragmentActivity」

```
package net.macdidi.atk

...
class MapsActivity : FragmentActivity(), ... {
    ...
}
```

完成修改以後執行應用程式，這是套用Material Design樣式後的畫面：



開啟「res/values/styles.xml(v21)」，參考下列的內容修改為亮色樣式：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">

    </style>
</resources>
```

這是套用Material Design亮色樣式後的畫面：



同樣在「res/values/styles.xml(v21)」，參考下列的內容修改為亮色樣式搭配暗色的功能表：

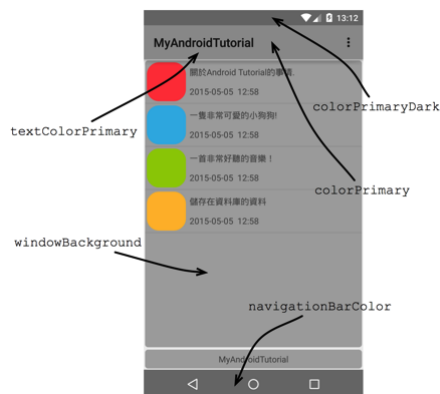
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">

    </style>
</resources>
```

這是套用Material Design亮色樣式搭配暗色功能表後的畫面：



Material Design還可以讓你調整樣式的設定，下面的圖型說明畫面的區域和相對的設定：



你可以依照應用程式的需求，加入自定的樣式設定，建立自定的畫面風格。同樣在「res/values/styles.xml(v21)」，參考下列修改這個樣式設定檔：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">
        <item name="android:colorPrimary">#999999</item>
        <item name="android:navigationBarColor">#777777</item>
    </style>
</resources>
```

這是套用自定樣式設定後的畫面：



18-2 Transition API

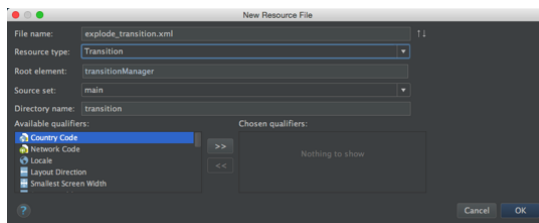
為了讓使用者在操作應用程式的時候，畫面可以有更好的回應與感受，開發人員通常會使用Android提供的動畫API，讓畫面可以反應使用者的操作。Material Design提供統一的規則與作法，讓使用者體驗的部份可以比較完整與容易實作，也不會造成不同應用程式之間的差異。

Material Design提供Transition API，讓應用程式在切換不同畫面元件的時候，可以容易設定轉換的效果。從Android 5 Lollipop API Level 21開始提供下列三種效果：

- **explode**：從畫面中央移入或移出。
- **slide**：從畫面兩側移入或移出。
- **fade**：淡入與淡出。

實作畫面轉換效果最方便的作法是設計轉換效果的資源，依照下列的步驟建立一個轉換效果資源：

1. 在「res」目錄按滑鼠右鍵後選擇「New -> Android Resource File」。
2. 在「File Name:」欄位輸入「explode_transition.xml」。
3. 「Resource Type:」選擇「Transition」。
4. 選擇「OK」按鈕。



依照下列的內容，修改預設的資源檔：

```
<?xml version="1.0" encoding="utf-8"?>
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android">
    <explode android:duration="500"/>
    <changeBounds/>
    <changeTransform/>
    <changeClipBounds/>
    <changeImageTransform/>
</transitionSet>
```

開啟「res/values/styles.xml(v21)」，參考下列的內容修改設定檔：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">
        <item name="android:colorPrimary">#999999</item>
        <item name="android:navigationBarColor">#777777</item>

        <item name="android:windowContentTransitions">true</item>

        <item name="android:windowEnterTransition">@transition/explode_transition</item>
        <item name="android:windowExitTransition">@transition/explode_transition</item>
    </style>
</resources>
```

因為Transition API在Android 5 Lollipop、API Level 21開始提供，為了相容舊的版本，需要執行裝置版本的判斷。開啟「MainActivity.kt」，加入下列兩個啟動Activity元件的函式：

```
package net.macdidi.atk

...

class MainActivity : Activity() {

    ...
```

```

        private fun startActivityForVersion(intent: Intent, requestCode: Int) {
            // 如果裝置的版本是LOLLIPOP
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                // 加入畫面轉換設定
                startActivityForResult(intent, requestCode,
                    ActivityOptions.makeSceneTransitionAnimation(
                        this@MainActivity).toBundle())
            } else {
                startActivityForResult(intent, requestCode)
            }
        }

        private fun startActivityForVersion(intent: Intent) {
            // 如果裝置的版本是LOLLIPOP
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                // 加入畫面轉換設定
                startActivity(intent,
                    ActivityOptions.makeSceneTransitionAnimation(
                        this@MainActivity).toBundle())
            } else {
                startActivity(intent)
            }
        }
    }
}

```

接下來依照應用程式的需求，修改啟動元件的敘述。同樣在「MainActivity.kt」，找到「processControllers」函式，參考下列式片段修改程式碼，這是執行修改記事的功能：

```

package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

    private fun processControllers() {
        val itemListener = AdapterView.OnItemClickListener {
            // position: 使用者選擇的項目編號，第一個是0
            _, _, position, _ ->

            val item = itemAdapter.getItem(position)

            if (selectedCount > 0) {
                processMenu(item)
                itemAdapter[position] = item
            } else {
                val intent = Intent(

```



```

        "net.macdidi.atk.EDIT_ITEM")
        intent.putExtra("position", position)
        intent.putExtra("net.macdidi.atk.Item", item)

        // 改為呼叫這個函式，依照版本啟動Activity元件
        startActivityForVersion(intent, 1)
    }
}

item_list.setOnItemClickListener = itemListener

...
}

...

}

```

同樣在「MainActivity.kt」，找到「clickPreferences」函式，參考下列的程式片段修改程式碼，這是執行設定的功能：

```

package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

    // 設定
    fun clickPreferences(item: MenuItem) {
        // 改為呼叫這個函式，依照版本啟動Activity元件
        startActivityForVersion(Intent(this, PrefActivity::class.java))
    }

    ...

}

```

同樣在「MainActivity.kt」，找到「clickMenuItem」函式，參考下列的程式片段修改程式碼，這是執行新增記事的功能：

```

package net.macdidi.atk

...

class MainActivity : Activity() {

```

```

...

fun clickMenuItem(item: MenuItem) {
    when (item.itemId) {
        ...
        R.id.add_item -> {
            val intent = Intent("net.macedidi.atk.ADD_ITEM")

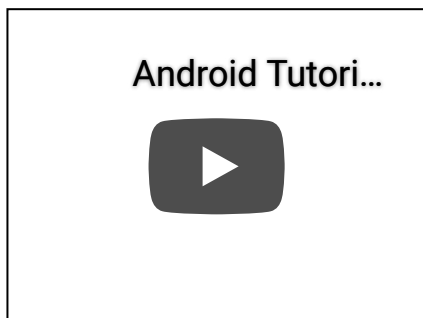
            // 改為呼叫這個函式，依照版本啟動Activity元件
            startActivityForVersion(intent, 0)
        }
        ...
    }
}

...

}

```

完成轉換效果資源與程式碼的修改，關於應用程式的部份（**AboutActivity**），因為使用對話框的樣式顯示，所以沒有加入轉換效果。這是執行應用程式以後的示範影片：



你可以依照應用程式的需求，選擇其它不同的轉換效果。使用上面說明的方式，建立另外一個名稱為「**fade_transition.xml**」檔案：

```

<?xml version="1.0" encoding="utf-8"?>
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android">
    <fade android:duration="500"/>
    <changeBounds/>
    <changeTransform/>
    <changeClipBounds/>
    <changeImageTransform/>
</transitionSet>

```

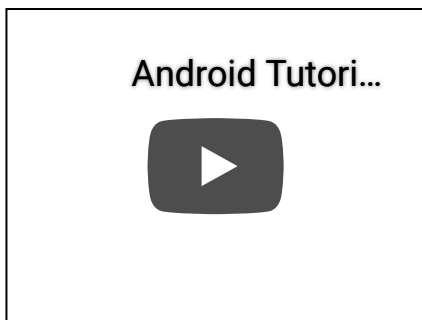
開啟「**res/values/styles.xml(v21)**」，參考下列的內容修改設定檔：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">
        <item name="android:colorPrimary">#999999</item>
        <item name="android:navigationBarColor">#777777</item>

        <item name="android:windowContentTransitions">true</item>

        <item name="android:windowEnterTransition">@transition/fade_transition</item>
        <item name="android:windowExitTransition">@transition/fade_transition</item>
    </style>
</resources>
```

完成以後執行應用程式，試試看不同的轉換效果。使用同樣的方式，也可以建立與使用「slide」轉換效果。這是執行應用程式的示範影片：



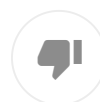
相關的檔案都可以在GitHub瀏覽與下載：

GitHub


<https://github.com/macdidi5/Android-Tutorial-Kotlin>

後續 >> Android Tutorial using Kotlin 第六堂 (2) Material Design – RecyclerView

Does Clearly work fine?



Shortcuts: **SHIFT+CTRL+C** to Toggle, **ESC** to Close.

 Give us feedback

Build upon ❤️ with Clearly

