

# Android Tutorial using Kotlin 第五堂

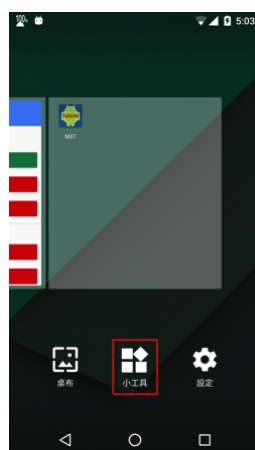
## （3）設計小工具元件 - AppWidget

[Android Tutorial using Kotlin 第五堂（2）系統通知服務 – Notification << 前情](#)

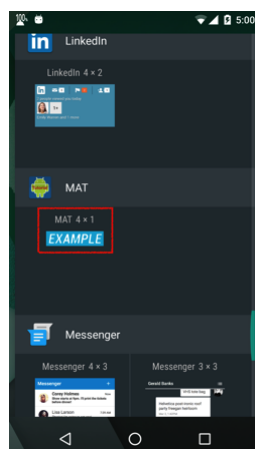
一般個人電腦或行動裝置的作業系統，可以在桌面上放置應用程式的捷徑，使用明顯的圖示和應用程式的名稱，讓使用者在桌面直接啟動常用的應用程式。一些使用者經常操作的功能，例如開啟或關閉裝置的網路或藍牙設備，如果可以不用啟動這些設定應用程式，使用者就可以在桌面上直接操作這些功能，那應該會比較方便一些。

Android平台提供一種特別的元件「AppWidget」，它可以讓使用者在桌面上直接瀏覽資料，或是執行一些簡單的操作。例如顯示時間、行事曆或氣候資訊，這種元件通常會把它稱為「小工具」元件。

這一章介紹設計AppWidget元件的作法，它的設計方式跟其它元件很不一樣。完成這一章的工作以後，為記事應用程式加入小工具元件。使用者在畫面長按以後，Android會開啟這樣的畫面，選擇「小工具」：



在選擇小工具的畫面，找到為記事應用應用程式設計好的元件，長按以後把它放到畫面指定的位置：



決定小工具的位置以後，元件自動開啟選擇記事的畫面，選擇其中一個記事項目：



畫面上就會放置一個顯示記事標題的小工具元件：



依照同樣的步驟，可以在畫面加入其它顯示記事標題的小工具：



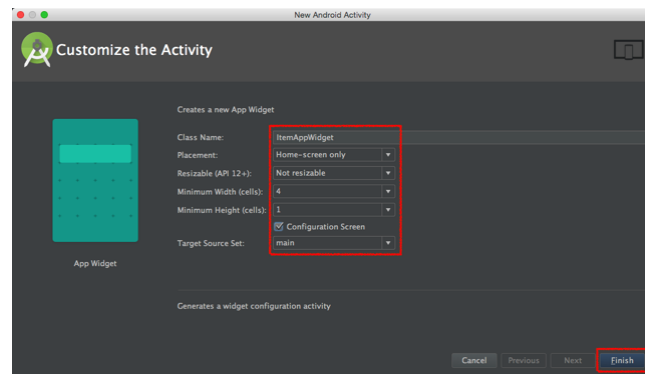
## 17-1 加入小工具元件

現在開始為記事應用程式加入小工具元件。啟動Android Studio與記事應用程式以後，在「app」目錄上按滑鼠右鍵 -> 選擇 -> Widget -> AppWidget」，依照下列的說明輸入需要的資訊：

- Class Name 輸入「ItemAppWidget」。
- Placement 選擇「Home-screen only」。
- Resizable(API 12+) 選擇「Not resizable」。
- Minimum Width (cells) 選擇「4」。

- Minimum Height (cells) 選擇「1」。
- 勾選「Configuration Screen」。

完成後選擇「Finish」按鈕：



Android Studio會建立許多小工具元件需要的程式碼與設定檔：

- **ItemAppWidget.kt**：小工具元件類別。
- **ItemAppWidgetConfigureActivity.kt**：小工具設定元件類別，選擇記事項目。
- **res/layout/itemappwidget.xml**：小工具元件使用的畫面資源。
- **res/layout/itemappwidget\_configure.xml**：小工具設定元件使用的畫面資源。
- **res/xml/itemappwidget\_info.xml**：小工具專用的設定檔。
- **res/drawable-nodpi/exampleappwidgetpreview.png**：在小工具選擇畫面顯示的縮圖。
- **AndroidManifest.xml**：自動加入小工具元件與小工具設定元件的設定。

## 17-2 實作小工具設定元件

使用者選擇記事小工具以後，必須先啟動選擇記事項目元件。小工具元件在專用的設定檔可以設定這個功能，開啟「res/xml/item\_app\_widget\_info.xml」，檢視裡面的內容：

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    android:configure：小工具設定元件類別
    android:initialLayout：小工具元件使用的畫面資源
    android:previewImage：縮圖
-->
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="net.macdidi.atk.ItemAppWidgetConfigureActivity"
    android:initialKeyguardLayout="@layout/item_app_widget"
    android:initialLayout="@layout/item_app_widget"
    android:minHeight="40dp"
    android:minWidth="250dp"
    android:previewImage="@drawable/example_appwidget_preview"
    android:updatePeriodMillis="86400000"
    android:widgetCategory="home_screen"/>
```

Android Studio會自動產生預設的設定元件與畫面資源，這個設定元件可以重複使用主畫面元件的畫面資源，所以刪除在「res/layout」下的「itemappwidget\_configure.xml」。接下來開啟「ItemAppWidgetConfigureActivity.kt」，先清除類別裡面的內容，然後加入下面的欄位變數宣告：

```
package net.macdidi.atk

...

class ItemAppWidgetConfigureActivity : Activity() {

    internal var mAppWidgetId = AppWidgetManager.INVALID_APPWIDGET_ID

    // 選擇小工具使用的記事項目
    private val item_list: ListView by bind(R.id.item_list)
    private val itemDAO: ItemDAO by lazy { ItemDAO(applicationContext) }
    private val items: ArrayList<Item> by lazy { itemDAO.all }
    private val itemAdapter: ItemAdapter
        by lazy { ItemAdapter(this, R.layout.single_item, items) }

    ...

}
```

同樣在「ItemAppWidgetConfigureActivity.kt」，加入下列的函式宣告：

```
package net.macdidi.atk

...

class ItemAppWidgetConfigureActivity : Activity() {

    ...

    companion object {

        private val PREFS_NAME = "net.macdidi.atk.ItemAppWidget"
        private val PREF_PREFIX_KEY = "appwidget_"

        // 儲存選擇的記事編號
        fun saveItemPref(context: Context, appWidgetId: Int, id: Long) {
            val prefs = context.getSharedPreferences(PREFS_NAME, 0).edit()
            prefs.putLong(PREF_PREFIX_KEY + appWidgetId, id)
            prefs.commit()
        }

        // 讀取記事編號
        fun loadItemPref(context: Context, appWidgetId: Int): Long {
```

```

        val prefs = context.getSharedPreferences(PREFS_NAME, 0)

        return prefs.getLong(PREF_PREFIX_KEY + appWidgetId, 0)
    }

    // 刪除記事編號
    fun deleteItemPref(context: Context, appWidgetId: Int) {
        val prefs = context.getSharedPreferences(PREFS_NAME, 0).edit()
        prefs.remove(PREF_PREFIX_KEY + appWidgetId)
        prefs.commit()
    }

}

// 選擇記事項目
internal var itemListener: AdapterView.OnItemClickListener =
    AdapterView.OnItemClickListener {
        _, _, position, _ ->
        val context = this@ItemAppWidgetConfigureActivity

        // 讀取與儲存選擇的記事物件
        val item = itemAdapter.getItem(position)
        saveItemPref(context, mAppWidgetId, item.id)

        val appWidgetManager = AppWidgetManager.getInstance(context)
        ItemAppWidget.updateAppWidget(
            context, appWidgetManager, mAppWidgetId)
        val resultValue = Intent()
        resultValue.putExtra(
            AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId)
        setResult(Activity.RESULT_OK, resultValue)

        finish()
    }
}

```

同樣在「ItemAppWidgetConfigureActivity.kt」，依照下面的內容加入「onCreate」函式：

```

package net.macdidi.atk

...

class ItemAppWidgetConfigureActivity : Activity() {

    ...

    public override fun onCreate(ifecycle: Bundle?) {
        super.onCreate(ifecycle)
    }
}

```

```

        setResult (Activity.RESULT_CANCELED)

        // 改為使用應用程式主畫面
        setContentView (R.layout.activity_main)

        // 建立與設定選擇小工具使用的記事項目需要的物件
        item_list.adapter = itemAdapter
        item_list.setOnItemClickListener = itemListener

        val extras = intent.extras

        if (extras != null) {
            mAppWidgetId = extras.getInt(
                AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID)
        }

        if (mAppWidgetId == AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish()
            return
        }
    }

    ...

}

```

### 17-3 實作小工具元件

接下來修改預設的小工具元件類別與畫面資源。先建立小工具使用的背景顏色資源，在「res/drawable」目錄上按滑鼠右鍵，「New -> Drawable resource file」，在「File name」輸入「*widgetdrawable*」後選擇「OK」，修改*widgetdrawable*為下面的容：

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <corners
        android:topLeftRadius="20sp"
        android:topRightRadius="20sp"
        android:bottomLeftRadius="20sp"
        android:bottomRightRadius="20sp" />

    <solid android:color="#1E88E5"/>

    <stroke android:color="#1976D2" android:width="3dp"/>

</shape>

```

開啟「res/layout/item\_app\_widget.xml」，這是小工具元件使用的畫面資源，決定小工具在畫面上的樣子。依照下面的內容做設定：

```
<!-- 修改背景 -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/widget_margin"
    android:background="@drawable/widget_drawable">

    <!-- 修改文字、大小、顏色與刪除背景 -->
    <TextView
        android:id="@+id/appwidget_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/app_name"
        android:textColor="#EEEEEE"
        android:textSize="16sp"
        android:textStyle="bold|italic"
        android:layout_margin="8dp"
        android:contentDescription="@string/app_name" />

</RelativeLayout>
```

開啟小工具元件類別「ItemAppWidget.kt」，找到「onDeleted」函式，依照下面的內容修改程式碼：

```
package net.macdidi.atk

...

class ItemAppWidget : AppWidgetProvider() {

    ...

    override fun onDeleted(context: Context, appWidgetIds: IntArray) {
        val N = appWidgetIds.size

        for (i in 0 until N) {
            // 刪除小工具已經儲存的記事編號
            ItemAppWidgetConfigureActivity.deleteItemPref(
                context, appWidgetIds[i])
        }
    }
}
```

```
...  
  
}
```

同樣在「ItemAppWidget.kt」，找到「updateAppWidget」函式，依照下面的內容修改程式碼：

```
package net.macdidi.atk  
  
...  
  
class ItemAppWidget : AppWidgetProvider() {  
  
    ...  
  
    companion object {  
  
        internal fun updateAppWidget(context: Context,  
                                      appWidgetManager: AppWidgetManager,  
                                      appWidgetId: Int) {  
  
            // 讀取小工具儲存的記事編號  
            val id = ItemAppWidgetConfigureActivity.loadItemPref(  
                context, appWidgetId)  
            // 建立小工具畫面元件  
            val views = RemoteViews(  
                context.packageName, R.layout.item_app_widget)  
            // 讀取指定編號的記事物件  
            val itemDAO = ItemDAO(context.applicationContext)  
            val item = itemDAO[id]  
  
            // 設定小工具畫面顯示記事標題  
            views.setTextViewText(R.id.appwidget_text,  
                item?.title ?: "NA")  
  
            // 點選小工具畫面的記事標題後，啟動記事應用程式  
            val intent = Intent(context, MainActivity::class.java)  
            val pending = PendingIntent.getActivity(  
                context, 0, intent, 0)  
            views.setOnClickPendingIntent(R.id.appwidget_text, pending)  
  
            // 更新小工具  
            appWidgetManager.updateAppWidget(appWidgetId, views)  
        }  
    }  
  
}
```



## 17-4 執行小工具與設定元件的設定

最後記得要在應用程式設定檔中，使用「**receiver**」標籤為小工具元件加入需要的設定。小工具設定元件也需要使用「**activity**」必要的設定。開啟「**AndroidManifest.xml**」檢視裡面設定的內容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.macdidi.atk">

    ...

    <application ...>
        ...

        <!-- 小工具元件 -->
        <receiver android:name=".ItemAppWidget">
            <!-- 一定要加入這個Action名稱的設定 -->
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <!-- 使用android:resource指定小工具專用設定檔的資源名稱 -->
            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/item_app_widget_info" />
        </receiver>

        <!-- 小工具設定元件 -->
        <activity android:name=".ItemAppWidgetConfigureActivity">
            <!-- 一定要加入這個設定 -->
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

完成這一章所有的功能了，執行應用程式，使用一開始介紹的方式，加入幾個記事小工具。

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（1）Material Design – Theme與Transition](#)

Does Clearly work fine?

Does Clearly work fine?



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

 Give us feedback

Build upon ♥ with Clearly