

Android Tutorial using Kotlin 第二堂

（4）建立與使用Activity元件

Android Tutorial using Kotlin 第二堂（3）應用程式與使用者的互動 << 前情

大部份的Android應用程式，都需要一些畫面提供使用者執行操作或瀏覽資料。Android系統使用Activity元件，負責提供應用個畫面的所有相關工作。一個畫面就是一個繼承自「android.app.Activity」的類別，所以通常會把它稱為Activity元件，也有「活動」元件。Activity元件幾乎是Android應用程式中最常使用的，應用程式的功能如果比較複雜，需要提供比較多的操作和資料的畫面，就會包含很多Activity元件。

每一個Activity元件除了撰寫需要的Kotlin原始程式碼，也需要在應用程式設定檔加入相關的設定，在application的開始和結束面，使用activity標籤為每一個Activity元件加入設定，所以從應用程式設定檔的內容，也可以知道一個應用程式有幾個Activity

這一章介紹Activity元件的開發與設定方式，並瞭解關於Activity元件的生命週期概念，還有Activity元件之間的互動與資料傳輸

8-1 記事本應用程式

之前已經建立好的應用程式主畫面，提供基本的資料瀏覽與操作功能，現在要為它加入兩個Activity元件，一個用來顯示關於I式的資訊，另一個用來新增一筆記事本資料。

依照之前的說明，為應用程式設計需要的Activity元件，應該要先規劃好畫面與資源的需求，而且也要想好操作的流程，所以簡單的畫一個像這樣的圖型：



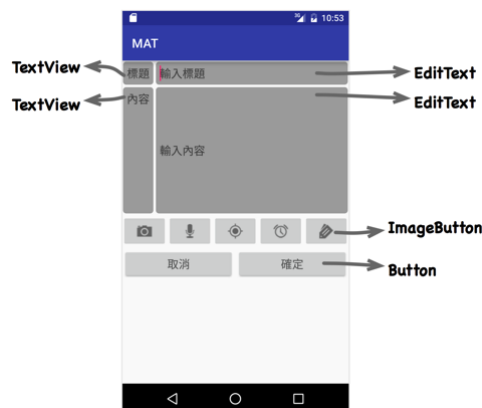
在規畫這些元件的時候，就可以整理好需要建立的Activity與畫面配置檔：

- 應用程式資訊：AboutActivity.kt，activity_about.xml
- 新增記事本：ItemActivity.kt，activity_item.xml

使用者點擊主畫面下方的應用程式名稱，應用程式啟動資訊元件，畫面的設計會比較簡單一些，只有兩個TextView和一個Button，畫面需要的文字資源也要先建立好。

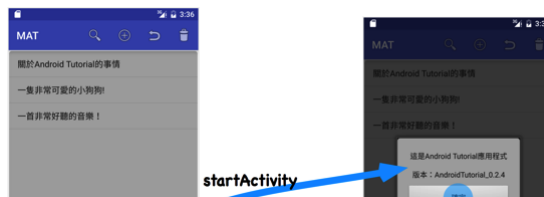


使用者選擇功能表的新增項目，應用程式啟動新增記事本元件，在這個畫面讓使用者輸入標題與內容，為了後續加入的功能，畫面中提供記事本功能按鈕，例如錄音與地圖。



8-2 建立與啟動Activity元件

現在開始建立顯示應用程式資訊的Activity元件，不過要先瞭解Activity元件的基本運作。應用程式可以呼叫「startActivity」函式其它Activity元件，呼叫「finish」函式可以結束Activity元件：



現在開始新增應用程式資訊元件，在Android Studio開啟MyAndroidStudio應用程式。開啟「res/values/strings.xml」，加入文字資源：

```
<string name="version">版本: AndroidTutorial_0.2.4</string>
```

開啟「res/values/colors.xml」，加入下列的顏色資源：

```
<color name="dark_text">#111111</color>
```

在最頂端的「app」目錄按滑鼠左鍵，選擇「New -> Activity -> Empty Activity」，元件與畫面配置檔名稱依照上面的規劃，稱為「AboutActivity」，畫面資源名為「activity_about」。開啟畫面配置檔「activity_about.xml」，修改為下面的內容：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/rectangle_drawable"
    tools:context="net.macedidi.atk.AboutActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_margin="@dimen/default_margin"
        android:padding="@dimen/default_padding"
        android:text="@string/about"
        android:textColor="@color/dark_text" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_margin="@dimen/default_margin"
        android:padding="@dimen/default_padding"
        android:text="@string/version"
        android:textColor="@color/dark_text" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_margin="@dimen/default_margin"
        android:padding="@dimen/default_padding"
        android:text="@android:string/ok"
        android:onClick="clickOk" />

</LinearLayout>
```

開啟「AboutActivity.kt」，加入取消應用程式標題的敘述，還有在負責執行按鈕工作的函式中加入結束Activity元件的敘述，以及它不需要的程式碼：

```

package net.macdidi.atk

import android.app.Activity
import android.os.Bundle
import android.view.View
import android.view.Window

// 從AppCompatActivity改為Activity
class AboutActivity : Activity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // 取消元件的應用程式標題
        requestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.activity_about)
    }

    // 結束按鈕
    fun clickOk(view: View) {
        // 呼叫這個函式結束Activity元件
        finish()
    }
}

```

Android應用程式的每一個Activity元件，都需要在應用程式設定檔加入對應的設定，使用Android Studio建立Activity元件會自動為你加入預設的設定。開啟「manifests/AndroidManifest.xml」，找到ADT為你加入的設定，把它改為下面的內容：

```

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- 關於應用程式的資訊 -->
        <!-- 因為使用對話框的樣式，所以不用設定標題 -->

```

```

        <activity
            android:name=".AboutActivity"
            android:theme="@android:style/Theme.Dialog" />
    </application>

</manifest>

```

因為這個Activity元件的內容比較簡單，使用整個螢幕顯示畫面的話，看起來會比較空曠一些，所以在設定檔加入「`android:theme="@android:style/Theme.Dialog"`」的設定，讓這個Activity元件使用對話框的樣式。

最後的工作就是執行啟動這個Activity元件，先檢查應用程式主畫面的畫面配置檔「`activity_main.xml`」，看看主畫面下方顯示程式名稱元件有沒有加入需要的設定：

```

<LinearLayout ...>
    ...
    <!-- 加入「android:clickable="true"」的設定，TextView元件才可以點擊 -->
    <!-- 加入「android:onClick="函式名稱"」的設定 -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_margin="@dimen/default_margin"
        android:padding="@dimen/default_padding"
        android:background="@drawable/rectangle_drawable"
        android:text="@string/app_name"
        android:clickable="true"
        android:onClick="aboutApp" />
</LinearLayout>

```

開啟「`MainActivity.kt`」，找到`aboutApp`函式，把程式碼改為下面的內容：

```

package net.macdidi.atk

import android.content.Intent
...

class MainActivity : AppCompatActivity() {

    ...

    // 函式名稱與onClick的設定一樣，參數的型態是android.view.View
    fun aboutApp(view: View) {
        // 建立啟動另一個Activity元件需要的Intent物件
        // 建構式的第一個參數：「this」
        // 建構式的第二個參數：「Activity元件類別名稱::class.java」
    }
}

```

```

        val intent = Intent(this, AboutActivity::class.java)
        // 呼叫「startActivity」，參數為一個建立好的Intent物件
        // 這行敘述執行以後，如果沒有任何錯誤，就會啟動指定的元件
        startActivity(intent)
    }

}

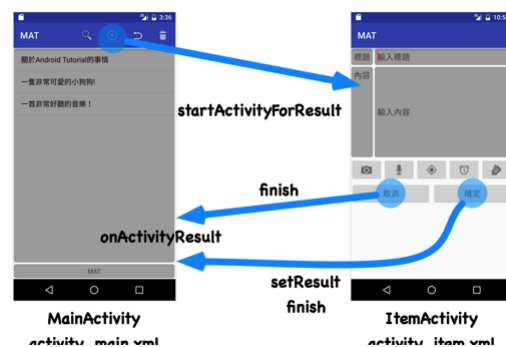
```

執行應用程式，看看點擊主畫面下方應用程式名稱元件後，會不會啟動與顯示新的畫面。在啟動的畫面點擊確定按鈕，應用程序會回到主畫面。這是在應用程式啟動與結束一個Activity元件的基本作法。

8-3 在結束Activity元件時傳送資料

在一般的應用程式運作的時候，經常需要啟動另一個Activity元件執行選擇、輸入或修改資料的功能，完成工作以後，再把資料給原來的Activity元件使用。以記事本應用程式來說，主畫面負責顯示所有的記事資料，需要新增資料的時候，啟動一個讓使用者輸入資料的Activity元件，完成新增的工作回到主畫面，這個新增的記事資料就要加入主畫面。

如果應用程式在啟動的Activity元件結束並返回後，還要執行一些特定的工作，就要使用「startActivityForResult」啟動Activity元件。這是新增記事本的元件流程：



決定應用程式的流程以後，現在開始設計新增記事用的Activity元件。在最頂端的「app」目錄按滑鼠左鍵，選擇「New -> Activity -> Empty Activity」，元件與畫面配置檔名稱依照上面的規劃，元件名稱為「ItemActivity」，畫面資源名稱為「activity_item」，開啟activity_item.xml，把它改為下面的內容：

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1"
    tools:context="net.macdidi.atk.ItemActivity">

    <TableRow>

        <TextView
            android:text="@string/title"
            android:background="@drawable/rectangle_drawable"
            android:padding="6sp"

```

```

        android:layout_margin="2sp"
        android:textAppearance="?android:attr/textAppearanceMedium" />

<EditText
    android:id="@+id/title_text"
    android:hint="@string/enter_title"
    android:background="@drawable/rectangle_drawable"
    android:padding="6sp"
    android:layout_margin="2sp"
    android:textAppearance="?android:attr/textAppearanceMedium" />
</TableRow>

<TableRow>

    <TextView
        android:text="@string/content"
        android:layout_height="200sp"
        android:layout_gravity="center_vertical"
        android:background="@drawable/rectangle_drawable"
        android:padding="6sp"
        android:layout_margin="2sp"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/content_text"
        android:hint="@string/enter_content"
        android:layout_gravity="top"
        android:layout_height="200sp"
        android:background="@drawable/rectangle_drawable"
        android:padding="6sp"
        android:layout_margin="2sp"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</TableRow>

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="*">

    <TableRow>

        <ImageButton
            android:id="@+id/take_picture"
            android:src="@drawable/take_picture_icon"
            android:onClick="clickFunction" />

        <ImageButton
            android:id="@+id/record_sound"
            android:src="@drawable/record_sound_icon"
            android:onClick="clickFunction" />

        <ImageButton

```

```

        android:id="@+id/set_location"
        android:src="@drawable/location_icon"
        android:onClick="clickFunction" />

<ImageButton
    android:id="@+id/set_alarm"
    android:src="@drawable/alarm_icon"
    android:onClick="clickFunction" />

<ImageButton
    android:id="@+id/select_color"
    android:src="@drawable/select_color_icon"
    android:onClick="clickFunction" />
</TableRow>
</TableLayout>

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="*">

    <TableRow>

        <Button
            android:id="@+id/cancel_item"
            android:text="@android:string/cancel"
            android:onClick="onSubmit"
            android:padding="6sp"
            android:layout_margin="2sp"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <Button
            android:id="@+id/ok_teim"
            android:text="@android:string/ok"
            android:onClick="onSubmit"
            android:padding="6sp"
            android:layout_margin="2sp"
            android:textAppearance="?android:attr/textAppearanceMedium" />
    </TableRow>
</TableLayout>

</TableLayout>

```

開啟「ItemActivity.kt」，修改為下面的內容。為了以後需要擴充的功能，加入一些控制按鈕執行工作的程式碼：

```

package net.macdidi.atk

import android.app.Activity
import android.os.Bundle

```



```

import android.support.v7.app.AppCompatActivity
import android.view.View
import android.widget.EditText

class ItemActivity : AppCompatActivity() {

    private val title_text : EditText by bind(R.id.title_text)
    private val content_text : EditText by bind(R.id.content_text)

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

    // 點擊確定與取消按鈕都會呼叫這個函式
    fun onSubmit(view: View) {
        // 確定按鈕
        if (view.id == R.id.ok_item) {
            // 讀取使用者輸入的標題與內容
            val titleText = title_text.text.toString()
            val contentText = content_text.text.toString()

            // 設定標題與內容
            intent.putExtra("titleText", titleText)
            intent.putExtra("contentText", contentText)

            // 設定回應結果為確定
            setResult(Activity.RESULT_OK, intent)
        } else {
            // 設定回應結果為取消
            setResult(Activity.RESULT_CANCELED, intent)
        }

        // 結束
        finish()
    }

    // 以後需要擴充的功能
    fun clickFunction(view: View) {
        when (view.id) {
            R.id.take_picture -> {
            }
            R.id.record_sound -> {
            }
            R.id.set_location -> {
            }
            R.id.set_alarm -> {
            }
            R.id.select_color -> {
            }
        }
    }
}

```

```
}
```

開啟「AndroidManifest.xml」，找到Android Studio為你加入的設定，把它改為下面的內容：

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".AboutActivity"
            android:theme="@android:style/Theme.Dialog" />

        <!-- 記事項目元件 -->
        <activity android:name=".ItemActivity" />

    </application>

</manifest>
```

開啟「MainActivity.kt」，把程式碼改為下面的內容：

```
package net.macdidi.atk

...

class MainActivity : AppCompatActivity() {

    ...

    // 使用者選擇所有的選單項目都會呼叫這個函式
    fun clickMenuItem(item: MenuItem) {
```

```

// 使用參數取得使用者選擇的選單項目元件編號
val itemId = item.itemId

// 判斷該執行什麼工作，目前還沒有加入需要執行的工作
when (itemId) {
    R.id.search_item -> {
    }
    // 使用者選擇新增選單項目
    R.id.add_item -> {
        // 建立啟動另一個Activity元件需要的Intent物件
        val intent = Intent(this, ItemActivity::class.java)
        // 呼叫「startActivityForResult」，第二個參數「0」目前沒有使用
        startActivityForResult(intent, 0)
    }
    R.id.revert_item -> {
    }
    R.id.delete_item -> {
    }
}
}
...
}

```

使用「startActivityForResult」啟動Activity元件，結束並返回以後，Android會呼叫「onActivityResult」函式一次。所以覆寫這個函式，在裡面執行需要的判斷與工作。同樣在「MainActivity.kt」，因為原來使用字串陣列提供資料給ListView元件，現在要把這個陣列換成「ArrayList」物件，這樣可以修改ListView包裝的資料項目。把程式碼改為下面的內容：

```

package net.macdidi.atk

...

class MainActivity : AppCompatActivity() {

    private val item_list : ListView by bind(R.id.item_list)
    private val show_app_name: TextView by bind(R.id.show_app_name)

    // 換掉原來的字串陣列
    private val data = ArrayList<String>()

    private val adapter : ArrayAdapter<String>
        by lazy {ArrayAdapter(this, android.R.layout.simple_list_item_1, data)}

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

        processControllers()
    }
}

```

```

        // 加入範例資料
        data.add("關於Android Tutorial的事情");
        data.add("一隻非常可愛的小狗狗!");
        data.add("一首非常好聽的音樂!");

        item_list!!.adapter = adapter
    }

    override fun onActivityResult(requestCode: Int,
                                    resultCode: Int, data: Intent) {
        // 如果被啟動的Activity元件傳回確定的結果
        if (resultCode == Activity.RESULT_OK) {
            // 讀取標題
            val titleText = data.getStringExtra("titleText")
            // 加入標題項目
            this.data.add(titleText)
            // 通知資料已經改變，ListView元件才會重新顯示
            adapter.notifyDataSetChanged()
        }
    }
}

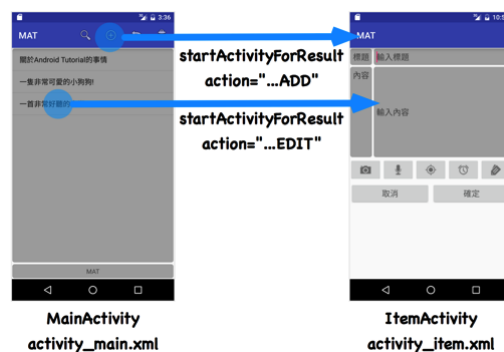
...
}

```

執行應用程式，點擊功能表的新增項目，在啟動的畫面輸入標題後，選擇確定按鈕，回到主畫面後，看看有沒有多一筆你剛新增的資料。

8-4 在啟動Activity元件時傳送資料

以這個記事應用程式來說，除了已經寫好的新增記事資料功能，通常也需要讓使用者修改記事資料。在主畫面點擊一筆需要修改記事項目以後，應用程式開啟修改記事的元件，讓使用者執行修改資料的工作。這個修改記事的元件其實跟新增記事的功能是共用的，所以通常就不會另外設計一個新的Activity元件，讓已經設計好的「ItemActivity」同時提供新增與修改兩種功能。這是新增與修改功能的流程：



為了讓一個Activity元件可以執行兩種工作，通常會幫這類元件另外取不同的「Action」名稱。開啟「AndroidManifest.xml」改為下面的內容：

```

<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application ... >
        ...
        <!-- 記事項目元件 -->
        <activity
            android:name=".ItemActivity">
            <intent-filter>
                <!-- 新增用的名稱 -->
                <action android:name="net.macdidi.myandroidtutorial.ADD_ITEM"/>
                <!-- 修改用的名稱 -->
                <action android:name="net.macdidi.myandroidtutorial.EDIT_ITEM"/>
                <!-- 一定要加入，內容固定不變 -->
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

開啟「ItemActivity.kt」，修改為下面的內容：

```

package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {

    private var title_text: EditText? = null
    private var content_text: EditText? = null

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

        // 如果是修改記事
        if (intent.action == "net.macdidi.atk.EDIT_ITEM") {
            // 接收與設定記事標題
            val titleText = intent.getStringExtra("titleText")
            title_text.setText(titleText)
        }
    }

    ...

}

```

開啟「MainActivity.kt」，把程式碼改為下面的內容：

```
package net.macdidi.atk

...

class MainActivity : AppCompatActivity() {

    private val item_list : ListView by bind(R.id.item_list)
    private val show_app_name: TextView by bind(R.id.show_app_name)

    // 換掉原來的字串陣列
    private val data = ArrayList<String>()

    private val adapter : ArrayAdapter<String>
        by lazy {ArrayAdapter(this, android.R.layout.simple_list_item_1, data)}

    ...

    override fun onActivityResult(requestCode: Int,
                                   resultCode: Int, data: Intent) {
        // 如果被啟動的Activity元件傳回確定的結果
        if (resultCode == Activity.RESULT_OK) {
            val titleText = data.getStringExtra("titleText")

            // 如果是新增記事
            if (requestCode == 0) {
                // 加入標題項目
                this.data.add(titleText)
                // 通知資料已經改變，ListView元件才會重新顯示
                adapter.notifyDataSetChanged()
            }
            // 如果是修改記事
            else if (requestCode == 1) {
                // 讀取記事編號
                val position = data.getIntExtra("position", -1)

                if (position != -1) {
                    // 設定標題項目
                    this.data[position] = titleText
                    // 通知資料已經改變，ListView元件才會重新顯示
                    adapter.notifyDataSetChanged()
                }
            }
        }
    }

    ...

    private fun processControllers() {
        // 建立選單項目點擊監聽物件
    }
}
```

```

        val itemListener = AdapterView.OnItemClickListener {
            // position: 使用者選擇的項目編號，第一個是0
            _, _, position, _ ->
            // 使用Action名稱建立啟動另一個Activity元件需要的Intent物件
            val intent = Intent("net.macdidi.atk.EDIT_ITEM")

            // 設定記事編號與標題
            intent.putExtra("position", position)
            intent.putExtra("titleText", data[position])

            // 呼叫「startActivityForResult」，第二個參數「1」表示執行修改
            startActivityForResult(intent, 1)
        }

        ...
    }

    ...

    fun clickMenuItem(item: MenuItem) {
        val itemId = item.itemId

        when (itemId) {
            R.id.search_item -> {
            }
            // 使用者選擇新增選單項目
            R.id.add_item -> {
                // 使用Action名稱建立啟動另一個Activity元件需要的Intent物件
                val intent = Intent("net.macdidi.atk.ADD_ITEM")
                // 呼叫「startActivityForResult」，第二個參數「0」表示執行新增
                startActivityForResult(intent, 0)
            }
            R.id.revert_item -> {
            }
            R.id.delete_item -> {
            }
        }
    }

    ...

}

```

完成這個階段的工作了，執行應用程式，試試看新增記事功能是否可以正常運作。在主畫面點選一個記事項目，修改記事標題後，看看主畫面的記事資料會不會顯示新增的記事項目。目前完成的功能並沒有處理記事資料的內容，也還沒有儲存到資料，所以不會保存新增與修改後的資料。

相關的檔案都可以在[GitHub](#)瀏覽與下載：




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

[後續 >> Android Tutorial using Kotlin 第三堂（1）為ListView元件建立自定畫面](#)

Does Clearly work fine?



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

 Give us feedback

Build upon ♥ with Clearly