

Android Tutorial using Kotlin 第六堂

(2) Material Design - RecyclerView

[Android Tutorial using Kotlin 第六堂 \(1\) Material Design – Theme與Transition << 前情](#)

Android在發表Android 5 Lollipop的同時，也發表Material Design和許多新的API，其中用來取代ListView元件的RecyclerView應用程式設計有比較大的影響。

一般的應用程式很常使用ListView元件，提供使用者一些資料的瀏覽與操作，它可以設計自己的項目畫面，加入需要的控制元。ListView元件可以應付大部份的需求。不過ListView元件在處理比較大量的資料時，效率就會比較差一些。另外需要設計比較符合使用者操作經驗的介面時，ListView元件的實作方式也會變的非常複雜。

從Android 5 Lollipop、API Level 21開始提供的RecyclerView元件，採用全新的Material Design設計，比ListView元件有更好的效能也更符合使用者的操作經驗。這一章把記事應用程式的主畫面元件，採用ListView元件實作的記事資料瀏覽與操作畫面，改為RecyclerView與Material Design的設計。接下來的修正幅度會比較大一些，所以會分為幾個階段，完成所有工作以後再執行測試。

把記事應用程式的主畫面改為RecyclerView以後，跟原來的樣子差不多，不過在完成應用程式以後，使用者操作的回應上就會不一樣了。

19-1 使用RecyclerView元件

RecyclerView元件在「android.support.v7.widget」套件下，應用程式需要使用RecyclerView元件，必須加入需要的設定。開啟「Gradle Scripts -> build.gradle (Module: app)」，參考下列的片段，在「dependencies」區塊加入需要的設定：

```
...
android {
    ...
}

dependencies {
    ...
    implementation 'com.android.support:recyclerview-v7:26.1.0'
    ...
}
```

加入上列的設定後，選擇功能表「Tools -> Android -> Sync Project with Gradle Files」，讓Android Studio執行相關的設定。

接下來修改應用程式的主畫面資源檔，開啟「res/layout/activity_main.xml」，參考下列的片段，把原來的ListView元件改為RecyclerView元件：

```

<LinearLayout ...>

    <!-- 使用RecyclerView元件 -->
    <android.support.v7.widget.RecyclerView
        android:id="@+id/item_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:layout_margin="@dimen/default_margin"
        android:dividerHeight="1sp"
        android:background="@drawable/rectangle_drawable"
        android:scrollbars="vertical" />

    <!-- 移除原來的ListView元件
    <ListView
        android:id="@+id/item_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:layout_margin="@dimen/default_margin"
        android:dividerHeight="1sp"
        android:background="@drawable/rectangle_drawable"
        android:divider="@color/divider_color" />
    -->

    ...

</LinearLayout>

```

開啟「res/values/dimens.xml」，加入下列的尺寸資源：

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    ...
    <dimen name="divider_size">2sp</dimen>
</resources>

```

為了讓畫面的操作可以符合RecyclerView與Material Design的設計，開啟「res/layout/single_item.xml」，參考下列的片段加的設定：

```

<?xml version="1.0" encoding="utf-8"?>
<!-- android:layout_height改為「wrap_content」 -->
<!-- 加入「android:background」的設定 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="?android:attr/selectableItemBackground">

        <RelativeLayout ... >

            ...

        </RelativeLayout>

    <LinearLayout ... >

        <TextView ... />

        <TextView ... />

        <!-- 加入分隔線，因為RecyclerView不能像ListView一樣設定項目分隔線 -->
        <View
            android:layout_width="match_parent"
            android:layout_height="@dimen/divider_size"
            android:layout_alignParentBottom="true"
            android:layout_marginRight="@dimen/divider_size"
            android:background="@color/divider_color"/>

    </LinearLayout>

</LinearLayout>

```

19-2 建立資料來源類別 – RecyclerView.Adapter

使用ListView元件時，通常會搭配「android.widget.ArrayAdapter」提供資料來源。RecyclerView元件使用「Adapter」為資料在作法上也有很大的差異。例如「RecyclerView.ViewHolder」的使用，ListView元件也可以使用ViewHolder包裝畫面元件，沒有強制規定，所以一般的作法都不會特別使用它。在實作提供給RecyclerView元件使用的資料來源時，就一定要使用RecyclerView.ViewHolder包裝每一個項目的畫面元件。

為了讓接下來的修改比較容易一些，保留ListView元件原來使用的「ItemAdapter」類別。在應用程式的主套件建立一個名稱「ItemAdapterRV」的類別，這是提供給RecyclerView元件使用的資料來源類別。參考下列的程式碼完成這個類別的實作：

```

package net.macdidi.atk

import android.graphics.drawable.GradientDrawable
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.RelativeLayout

```

```

import android.widget.TextView

open class ItemAdapterRV(private val items: MutableList<Item>)
    : RecyclerView.Adapter<ItemAdapterRV.ViewHolder>() {

    override fun onCreateViewHolder(
        parent: ViewGroup, viewType: Int): ItemAdapterRV.ViewHolder {
        val v = LayoutInflater.from(parent.context).inflate(
            R.layout.single_item, parent, false)

        return ViewHolder(v)
    }

    override fun onBindViewHolder(holder: ItemAdapterRV.ViewHolder, position: Int) {
        val item = items[position]

        // 設定記事顏色
        val background = holder.typeColor.background as GradientDrawable
        background.setColor(item.color.parseColor())

        // 設定標題與日期時間
        holder.titleView.text = item.title
        holder.dateView.text = item.localeDatetime

        // 設定是否已選擇
        holder.selectedItem.visibility =
            if (item.isSelected) View.VISIBLE else View.INVISIBLE
    }

    override fun getItemCount(): Int {
        return items.size
    }

    fun add(item: Item) {
        items.add(item)
        notifyItemInserted(items.size)
    }

    // 一定要使用ViewHolder包裝畫面元件
    inner class ViewHolder(var rootView: View)
        : RecyclerView.ViewHolder(rootView) {

        var typeColor: RelativeLayout = itemView.findViewById(R.id.type_color)
        var selectedItem: ImageView = itemView.findViewById(R.id.selected_item)
        var titleView: TextView = itemView.findViewById(R.id.title_text)
        var dateView: TextView = itemView.findViewById(R.id.date_text)

    }
}

```

19-3 完成主畫面元件

接下來剩下主畫面元件的修正工作，因為記事資料從ListView換成RecyclerView元件，有許多需要修改的地方，你也可以經由修過程，瞭解它們的差異。開啟「**MainActivity**」，參考下列的程式片段，修改下列的欄位變數：

```
package net.macdidi.atk

...

class MainActivity : Activity() {

    // 移除原來的ListView元件
    //private val item_list : ListView by bind(R.id.item_list)

    // 加入下列需要的元件
    private val item_list: RecyclerView by bind(R.id.item_list)
    private lateinit var itemAdapter: RecyclerView.Adapter<ItemAdapterRV.ViewHolder>
    private val rvLayoutManager: RecyclerView.LayoutManager
        by lazy { LinearLayoutManager(this) }

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

    // 移除原來的ItemAdapter
    //private val itemAdapter: ItemAdapter
    //        by lazy { ItemAdapter(this, R.layout.single_item, items) }

    ...

}
```

完成上列的修改以後，程式碼會產生很多錯誤，接下來會分成幾個段落完成所有修改的工作。首先找到「**onCreate**」方法，參閱下列的程式片段，依照註解的說明執行修改的工作：

```
package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

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

        // 移除註冊監聽事件的工作，要移到下面執行
        //processControllers()
    }
}
```

```

        if (itemDAO.count == 0) {
            itemDAO.createSampleData()
        }

        items.addAll(itemDAO.all)

        // 移除原來ListView元件執行的工作
        //item_list.adapter = itemAdapter

        // 執行RecyclerView元件的設定
        item_list.setHasFixedSize(true)
        item_list.layoutManager = rvLayoutManager

        // 在這裡執行註冊監聽事件的工作
        processControllers()
    }

    ...

}

```

同樣在「MainActivity」，找到「processControllers」方法，先移除方法中所有的程式碼，參考下列程式片段完成這個方法的

```

package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

    private fun processControllers() {

        // 實作ItemAdapterRV類別，加入註冊監聽事件的工作
        itemAdapter = object : ItemAdapterRV(items) {
            override fun onBindViewHolder(holder: ItemAdapterRV.ViewHolder,
                                           position: Int) {
                super.onBindViewHolder(holder, position)

                // 建立與註冊項目點擊監聽物件
                holder.rootView.setOnClickListener {
                    // 讀取選擇的記事物件
                    val item = items[position]

                    // 如果已經有勾選的項目
                    if (selectedCount > 0) {
                        // 處理是否顯示已選擇項目
                        processMenu(item)
                    }
                }
            }
        }
    }
}

```

```

        // 重新設定記事項目
        items[position] = item
    } else {
        val intent = Intent(
            "net.macdidi.atk.EDIT_ITEM")

        // 設定記事編號與記事物件
        intent.putExtra("position", position)
        intent.putExtra("net.macdidi.atk.Item", item)

        // 依照版本啟動Activity元件
        startActivityForVersion(intent, 1)
    }
}

// 建立與註冊項目長按監聽物件
holder.rootView.setOnLongClickListener {
    // 讀取選擇的記事物件
    val item = items[position]
    // 處理是否顯示已選擇項目
    processMenu(item)
    // 重新設定記事項目
    items[position] = item
    true
}

}

// 設定RecyclerView使用的資料來源
item_list.adapter = itemAdapter
}

...

}

```

同樣在「MainActivity」，找到「processMenu」方法，參考下列程式片段，在方法的最後加入需要的敘述：

```

package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

    private fun processMenu(item: Item?) {
        ...
    }
}

```

```

        // 通知項目勾選狀態改變
        itemAdapter.notifyDataSetChanged()
    }

    ...

}

```

同樣在「MainActivity」，找到「clickMenuItem」方法，參考下列的程式片段，依照註解的說明執行修改的工作：

```

package net.macdidi.atk

...

class MainActivity : Activity() {

    ...

    fun clickMenuItem(item: MenuItem) {
        when (item.itemId) {
            ...
            R.id.revert_item -> {
                // 改為使用items物件
                for (i in 0 until items.size) {
                    val ri = items[i]

                    if (ri.isSelected) {
                        ri.isSelected = false
                        // 移除
                        //items[i] = ri
                    }
                }

                selectedCount = 0
                processMenu(null)
            }
            R.id.delete_item -> {
                if (selectedCount == 0) {
                    return
                }

                val d = AlertDialog.Builder(this)
                val message = getString(R.string.delete_item)
                d.setTitle(R.string.delete)
                    .setMessage(String.format(message, selectedCount))
                d.setPositiveButton(android.R.string.yes) { _, _ ->
                    // 改為使用items物件
                    var index = items.size - 1

```



```

        while (index > -1) {
            val item = items[index]

            if (item.isSelected) {
                items.remove(item)
                // 刪除資料庫中的記事資料
                itemDAO.delete(item.id)
            }

            index--
        }

        // 移除
        //itemAdapter.notifyDataSetChanged()
        selectedCount = 0
        processMenu(null)
    }
    d.setNegativeButton(android.R.string.no, null)
    d.show()
}
}
}

...

}

```

完成所有修改的工作了，你可以認識RecyclerView的實作方式，比較原來使用ListView元件的作法，RecyclerView會比較簡化一些且不用撰寫很多程式碼，就可以提供Material Design的設計。

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 (3) Material Design – Shared Element與Floating Action Button

Does Clearly work fine?



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

 Give us feedback

Build upon  with Clearly

