

Kotlin 3

111360138 電子三甲 蔣安聖

讀書會

組員：

111360127 林煒哲

111360137 許鎧晏

111360138 蔣安聖

討論時間：2024/11/19 中午 12 點

地點：科研大樓



詢問 GPT 問題

1. `val array = resources.obtainTypedArray(R.array.image_list)` 是怎麼運作的？



在你的程式碼中，`resources.obtainTypedArray(R.array.image_list)` 是用來取得定義於 `res/values` 資源檔內的一個 `TypedArray`，這個 `TypedArray` 通常包含多個資源項目，例如圖片的資源 ID。

詳細解析

1. `R.array.image_list` 是什麼？

- `R.array.image_list` 是在 `res/values/arrays.xml` 文件中定義的資源 ID，通常是一個 `array` 標籤。
- 例如，`arrays.xml` 的內容可能如下：

```
xml
<resources>
  <array name="image_list">
    <item>@drawable/apple</item>
    <item>@drawable/banana</item>
    <item>@drawable/orange</item>
  </array>
</resources>
```

複製程式碼

- 這是一個資源陣列，包含三個圖片檔的資源 ID。

2. `obtainTypedArray` 的作用

- `resources.obtainTypedArray(R.array.image_list)` 的作用是從資源中取得這個陣列，並以 `TypedArray` 的形式返回。
- `TypedArray` 是 Android 提供的一個類別，用來處理資源陣列，它允許我們在程式中方便地存取陣列內的每一項。

3. 如何使用 `TypedArray`

- 取得 `TypedArray` 後，可以用以下方法存取其內容：
 - `getResourceId(index, defaultValue)`：獲取指定位置的資源 ID。
 - `getDrawable(index)`：獲取指定位置的圖片資源。
 - `length()`：獲取陣列的長度。
- 範例：

```
kotlin
val array = resources.obtainTypedArray(R.array.image_list)
for (index in 0 until array.length()) {
    val imageResId = array.getResourceId(index, 0) // 獲取每個圖片資源 ID
    // 使用這些資源 ID 來設定圖片或執行其他操作
}
array.recycle() // 用完後記得釋放資源
```

複製程式碼

4. 為什麼要使用 `recycle()`

- `TypedArray` 是一種有限資源，用完後應該釋放。
- 如果不使用 `recycle()`，可能導致記憶體洩漏或資源枯竭問題。

5. 結合你的程式碼的運作

- 在程式中，你使用了 `obtainTypedArray` 來取得圖片資源陣列，然後在 `for` 迴圈內：
 1. 使用 `getResourceId(index, 0)` 取得每張圖片的資源 ID。
 2. 建立 `Item` 資料物件，將圖片資源 ID (`photo`)、水果名稱 (`name`)、價格 (`price`) 添加進陣列。
- 程式的目的是將圖片資源與其他資訊組合成結構化的物件，用於 `GridView` 和 `ListView` 的顯示。

總結，`obtainTypedArray` 是一個方便的方法，用來從 XML 中獲取資源陣列，並透過其方法（如 `getResourceId`）輕鬆存取資源。

檔案中的 `image array` 並不叫 `array.xml` 而是叫 `strings.xml`，檔案名稱差別不大

2. 通訊錄中的 Myadapter 是如何運作的？



MyAdapter 是你在 **RecyclerView** 中使用的自訂適配器，它負責將資料與視圖連結起來，並處理聯絡人資料的顯示、刪除等操作。以下是 **MyAdapter** 的詳細運作邏輯解析：

MyAdapter 繼承自 RecyclerView.Adapter

MyAdapter 繼承了 **RecyclerView.Adapter<MyAdapter.ViewHolder>**，這樣它就成為了一個專門處理 **RecyclerView** 顯示資料的適配器。

1. ViewHolder 類別

ViewHolder 是一個靜態內部類別，負責為每個 **RecyclerView** 項目（也就是每個聯絡人資料）儲存與視圖相關的元件。在 **ViewHolder** 中，會綁定布局中的視圖元件，並提供方法來綁定資料。

ViewHolder 的運作

- 元件儲存：**ViewHolder** 儲存了布局中的 **TextView** 和 **ImageView**，這些元件將顯示聯絡人資料（姓名、電話）和刪除按鈕。

```
kotlin

private val tvName: TextView = v.findViewById(R.id.tvName)
private val tvPhone: TextView = v.findViewById(R.id.tvPhone)
private val imgDelete: ImageView = v.findViewById(R.id.imgDelete)
```

複製程式碼

- 資料綁定與監聽：**bind(item: Contact, clickListener: (Contact) -> Unit)** 方法將 **Contact** 資料綁定到相對應的視圖元件上，並設定刪除按鈕的點擊監聽器。

```
kotlin

fun bind(item: Contact, clickListener: (Contact) -> Unit) {
    tvName.text = item.name
    tvPhone.text = item.phone
    imgDelete.setOnClickListener {
        clickListener.invoke(item)
    }
}
```


複製程式碼

- bind** 方法會顯示 **Contact** 類別的 **name** 和 **phone** 屬性。
- 點擊 **imgDelete** 圖片時會觸發 **clickListener**，並傳回該項目（聯絡人資料），這是用來觸發刪除操作的。

2. onCreateViewHolder

onCreateViewHolder() 方法會在 **RecyclerView** 需要顯示新的項目時被調用，負責創建一個 **ViewHolder** 實例並返回。這個方法會通過 **LayoutInflater** 將 **adapter_row.xml** 布局檔案轉換為 **View**，並將其包裝在 **ViewHolder** 中。

kotlin

 複製程式碼


```
override fun onCreateViewHolder(viewGroup: ViewGroup, position: Int): ViewHolder {  
    val v = LayoutInflater.from(viewGroup.context)  
        .inflate(R.layout.adapter_row, viewGroup, false)  
    return ViewHolder(v)  
}
```

- `LayoutInflater.from(viewGroup.context).inflate(R.layout.adapter_row, viewGroup, false)` 會將 `adapter_row.xml` 佈局檔案轉換為 `View`，這個 `View` 代表 `RecyclerView` 中的一個項目。
- `ViewHolder(v)` 則包裝該 `View`，並返回 `ViewHolder` 實例。

3. onBindViewHolder

`onBindViewHolder()` 會將資料綁定到 `ViewHolder` 上，它會在 `RecyclerView` 顯示每一個項目時被呼叫。

kotlin

 複製程式碼


```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    holder.bind(data[position]) { item ->  
        // 刪除操作  
        data.remove(item)  
        notifyDataSetChanged()  
    }  
}
```

- `data[position]` 取出該位置的聯絡人資料，並將其傳遞給 `ViewHolder` 的 `bind` 方法。
- `bind` 方法會顯示該聯絡人的資料（姓名和電話），並且為刪除按鈕設定監聽器。
- 在 `clickListener.invoke(item)` 中，會從 `data` 中移除該項資料並調用 `notifyDataSetChanged()`，通知 Adapter 刷新 `RecyclerView`。

4. getItemCount

`getItemCount()` 方法返回資料列表的大小，`RecyclerView` 會根據這個數字來決定要顯示多少項目。

kotlin

 複製程式碼

```
override fun getItemCount() = data.size
```



- `data.size` 會回傳 `contacts` 的大小，也就是目前資料列表中聯絡人的數量。

總結：

1. 資料管理： `MyAdapter` 管理了一個 `ArrayList<Contact>` 資料列表，這些資料用來顯示在 `RecyclerView` 上。
2. 資料顯示：在 `onCreateViewHolder` 中創建每個項目的視圖，並在 `onBindViewHolder` 中將資料綁定到視圖上。
3. 刪除操作：每個項目有一個刪除按鈕，點擊該按鈕會觸發 `clickListener`，並從資料列表中移除該項資料。
4. 效能更新：當資料發生變動（例如刪除聯絡人）時，會調用 `notifyDataSetChanged()` 更新 `RecyclerView` 顯示。

這樣一來，你的 `MyAdapter` 就能根據 `contacts` 中的資料動態顯示聯絡人，並支持刪除操作。如果你對 `RecyclerView` 或 `Adapter` 還有其他疑問，隨時告訴我！

3. 體位檢測機中的 Thread 模擬檢測過程，是怎麼執行的？



這段程式碼的主要功能是執行一個模擬的進度條更新過程，然後計算並顯示 BMI、標準體重和體脂肪的結果。以下是整個程式執行的詳細說明：

1. 初始化 UI

在執行計算前，程式會初始化 UI 的內容：

- 將結果文字（ `tvWeightResult` , `tvFatResult` , `tvBmiResult` ）設置為預設值（如 無 ）。
- 將進度條（ `progressBar` ）的進度設為 0，並設置進度文字為 0% 。
- 顯示進度條的容器（ `llProgress` ）。

2. 執行緒的啟動

程式啟動了一個新的執行緒（ `Thread` ），用於模擬計算進度並避免在主執行緒（ `UI Thread` ）中執行阻塞操作。


3. 模擬進度更新

在執行緒中，執行以下操作：

1. 初始化進度：`progress = 0`。
2. 進度模擬：
 - 使用 `while` 迴圈，每次遞增進度直到達到 100。
 - 每次遞增時，使用 `Thread.sleep(50)` 延遲 50 毫秒，模擬計算過程。
3. 更新 UI：
 - 在迴圈中，調用 `runOnUiThread` 將進度更新到主執行緒，確保進度條和進度文字的即時更新。

4. 計算 BMI、標準體重和體脂肪

當進度條模擬完成後，執行實際的計算：

1. 取得輸入值：
 - `edHeight`：身高（公分）。
 - `edWeight`：體重（公斤）。
 - `edAge`：年齡（歲）。
2. 計算 BMI：
 - 使用公式：`BMI = 體重 ÷ (身高/100)^2`。
3. 計算標準體重與體脂肪：
 - 根據使用者性別（由 `btnBoy.isChecked` 確定）選擇不同公式：
 - 男性標準體重：`(身高 - 80) × 0.7`
 - 女性標準體重：`(身高 - 70) × 0.6`
 - 體脂肪公式考慮了 BMI 和年齡。

5. 更新最終結果

計算完成後，程式再次調用 `runOnUiThread` 切換到主執行緒，進行以下更新：

1. 隱藏進度條的容器（`llProgress`）。
 2. 顯示 BMI、標準體重和體脂肪的計算結果，並格式化成兩位小數。
-

心得

這次的實作讓我對 Android 開發的基礎功能有了更全面的了解，特別是在清單元件與非同步執行的部分收穫頗多。在清單元件中，我學會了如何透過 `ArrayAdapter` 以及自訂的布局來實現更靈活的顯示。在非同步執行的章節中，龜兔賽跑的 Lab 讓我直觀地體會到多線程的操作邏輯，在實際開發 APP 中，不可能不用多線程，因為很多情況是一段程式不需要太多互動但卻會占用大量時間，若不用多線程則畫面會被該程式卡住，使得程式運作極度不流暢，並產生許多問題，因此我覺得多線程是這次 Lab 中最重要的部分。

GitHub 程式連結與截圖

<https://github.com/DolphinBlast/JavaHW07>

