

讀書會

- 組員：龍品勳、張耀文、時顯文
- 討論時間：2024/11/18 10:00 AM
- 地點：圖書館

照片：



心得報告

- 心得： 這三個 Lab 越來越有設計手機應用軟體的感覺了。
- ChatGPT：

1. 請問這段程式碼的意思

```
setContentView(R.layout.activity_main)
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
insets }
```



這段程式碼主要是在 Android 中設定視圖佈局，並處理窗口插入 (Window Insets) 的情況。以下是逐步解釋：

1. setContentView(R.layout.activity_main)

這行是用來設定這個 Activity 的主佈局檔案。

`R.layout.activity_main` 指的是 `res/layout/activity_main.xml` 中的佈局檔案。

2. ViewCompat.setOnApplyWindowInsetsListener

這行設定了一個監聽器，專門處理窗口插入 (Window Insets) 。

窗口插入是指系統元件 (例如狀態欄、導航欄) 對應用視圖的影響。當這些系統元素出現或改變時，這個監聽器會被觸發。

```
findViewById(R.id.main)
```

這裡是找到佈局中的一個視圖元件，它的 ID 是 `main` 。

3. Lambda 表達式部分

kotlin

複製程式碼

```
{ v, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
    insets
}
```

(1) insets.getInsets(WindowInsetsCompat.Type.systemBars())

這行從 `insets` 中提取了 **系統欄 (system bars)** 的邊距 (插入區域)。
這包括了狀態欄 (Status Bar)、導航欄 (Navigation Bar) 等。

(2) v.setPadding(...)

將提取的系統欄邊距作為 **視圖的內邊距 (padding)** 設定。
這樣可以讓內容避免與狀態欄或導航欄重疊，從而提供更好的佈局體驗。

- `systemBars.left` : 左邊距。
- `systemBars.top` : 上邊距 (通常是狀態欄的高度)。
- `systemBars.right` : 右邊距。
- `systemBars.bottom` : 下邊距 (通常是導航欄的高度)。

(3) insets

最後返回 `insets`，表明已處理完插入區域。

用途

這段程式碼通常用於支持 **沉浸式佈局** 或處理 **全屏顯示** 的情境，確保應用視圖的佈局不會被系統元件遮擋。

總結

這段程式碼的核心作用是讓指定的視圖 (ID 為 `main`) 在系統欄 (例如狀態欄和導航欄) 出現時，自動調整其內邊距，從而防止視圖內容與系統欄重疊，提供更好的使用者體驗。

2. 請問這段程式碼的意思

```
private val startForResult =
registerForActivityResult( ActivityResultContracts.StartActivityForResult() ) { result:
ActivityResult -> if (result.resultCode == Activity.RESULT_OK) { val intent = result.data
val name = intent?.getStringExtra("name") ?: "" val phone =
```

```
intent?.getStringExtra("phone") ?: "" contacts.add(Contact(name, phone))  
myAdapter.notifyDataSetChanged() } }
```



這段程式碼使用了 Android Jetpack 的 Activity Result API，目的是在 Activity 之間傳遞結果，以下是詳細解析：

1. `registerForActivityResult`

`registerForActivityResult` 是一個新的 API，用於處理 Activity 之間的結果傳遞。

它替代了舊式的 `startActivityForResult`，提供了更簡潔且生命周期安全的方式來處理結果回傳。

- 參數 1：`ActivityResultContracts.StartActivityForResult()`
表示你要啟動一個新的 Activity 並希望從中獲取結果。
- 參數 2：Lambda 表達式，這裡是用來處理結果的邏輯。

2. Lambda 表達式解讀

kotlin

複製程式碼

```
{ result: ActivityResult ->  
    if (result.resultCode == Activity.RESULT_OK) {  
        val intent = result.data  
        val name = intent?.getStringExtra("name") ?: ""  
        val phone = intent?.getStringExtra("phone") ?: ""  
        contacts.add(Contact(name, phone))  
        myAdapter.notifyDataSetChanged()  
    }  
}
```

(1) `result: ActivityResult`

- `ActivityResult` 是回傳的結果類型，包含兩個重要屬性：
 - `result.resultCode`：返回的結果代碼，用於判斷操作是否成功。
 - `result.data`：返回的 `Intent`，包含傳遞回來的數據。

(2) `if (result.resultCode == Activity.RESULT_OK)`

- 這段判斷是檢查返回的結果代碼是否為 `RESULT_OK`，表示操作成功。

(3) `val intent = result.data`

- 取得回傳的 `Intent`，其中可能包含了其他數據。

(4) `intent?.getStringExtra("name") ?: ""`

- 從 `Intent` 中提取 `"name"` 和 `"phone"` 字串：
 - 如果數據存在，提取值。
 - 如果數據為 `null`，返回空字串 `""` 作為預設值。

(5) `contacts.add(Contact(name, phone))`

- 創建一個新的 `Contact` 對象，將提取到的 `name` 和 `phone` 添加到 `contacts` 列表中。

(6) `myAdapter.notifyDataSetChanged()`

- 通知資料適配器 (`RecyclerView.Adapter` 或其他適配器) 數據已更新，以便刷新 UI。
-

用途


這段程式碼的核心功能是：

1. 開啟一個新的 Activity，並等待它返回結果。
2. 根據返回的數據，更新聯絡人的資料列表 (`contacts`)。
3. 通知適配器刷新界面顯示。

使用方式示例

在主 Activity 中使用：


kotlin

 複製程式碼

```
val intent = Intent(this, SecondActivity::class.java)
startForResult.launch(intent)
```

在第二個 Activity 中回傳結果：

kotlin

 複製程式碼

```
val resultIntent = Intent()
resultIntent.putExtra("name", "John Doe")
resultIntent.putExtra("phone", "123456789")
setResult(Activity.RESULT_OK, resultIntent)
finish()
```

優點

- **安全**：與舊式 API 不同，不會導致內存泄漏或生命周期問題。
- **簡潔**：將結果處理邏輯與 Activity 實例綁定，代碼更清晰。

3. 請問 `thread` 在 Android 中的用途



在 Android 中，`Thread` 是 Java 的基本多執行緒工具，用於在應用中執行多執行緒任務。它的主要用途是執行耗時的任務，從而避免阻塞主執行緒（UI 執行緒），以確保應用的流暢性和響應性。

以下是 `Thread` 在 Android 中的常見用途：

1. 執行耗時任務

例如網路請求、文件讀寫、數據庫操作等。如果在主執行緒（UI 執行緒）中執行這些任務，可能會導致應用程式的界面無法響應，甚至觸發 ANR（Application Not Responding）。

示例：

kotlin

複製程式碼

```
Thread {  
    // 執行耗時操作，例如網路請求  
    val result = performNetworkOperation()  
  
    // 回到主執行緒更新 UI  
    runOnUiThread {  
        textView.text = result  
    }  
}.start()
```

2. 分離 UI 執行緒和工作執行緒

- **UI 執行緒**：處理用戶界面的繪製和交互，對用戶操作作出快速響應。
- **工作執行緒**：處理與界面無關的計算或後台任務，避免主執行緒的阻塞。

3. 併發處理

`Thread` 可以用於同時執行多個任務，提高應用的運行效率。例如：

- 同時下載多個文件。
- 從多個數據源並行讀取數據。

示例：

kotlin

複製程式碼

```
val thread1 = Thread { downloadFile("file1") }
val thread2 = Thread { downloadFile("file2") }

thread1.start()
thread2.start()
```


4. 背景處理


許多後台任務需要長時間運行，可以使用 `Thread` 處理這些任務，例如：

- 監控某些數據變化。
 - 長時間的文件壓縮或加密操作。
-

5. 延時操作

利用 `Thread.sleep()` 來實現延時效果。例如：

kotlin

 複製程式碼

```
Thread {  
    Thread.sleep(3000) // 延時 3 秒  
    runOnUiThread {  
        textView.text = "操作完成"  
    }  
}.start()
```

Android 的替代選擇

在 Android 中，`Thread` 是基礎工具，但不一定是最好的選擇，以下是更高級的選擇：

1. Handler 和 Looper

- 用於處理執行緒間的通信。
- 示例：在子執行緒中執行任務，然後將結果傳回主執行緒更新 UI。

2. AsyncTask (已廢棄)

- 適合處理輕量級的異步任務，但由於生命周期問題已被廢棄。

3. Executor 和 ExecutorService

- 更高效的執行緒池管理，避免創建過多執行緒導致的性能問題。

4. WorkManager

- 用於持久的背景任務，適合需要定時或持續執行的任務。

5. Coroutines (Kotlin 協程)

- 當前推薦的解決方案，簡化了異步任務的寫法，提供了比 `Thread` 更強大的功能。

協程示例：

kotlin

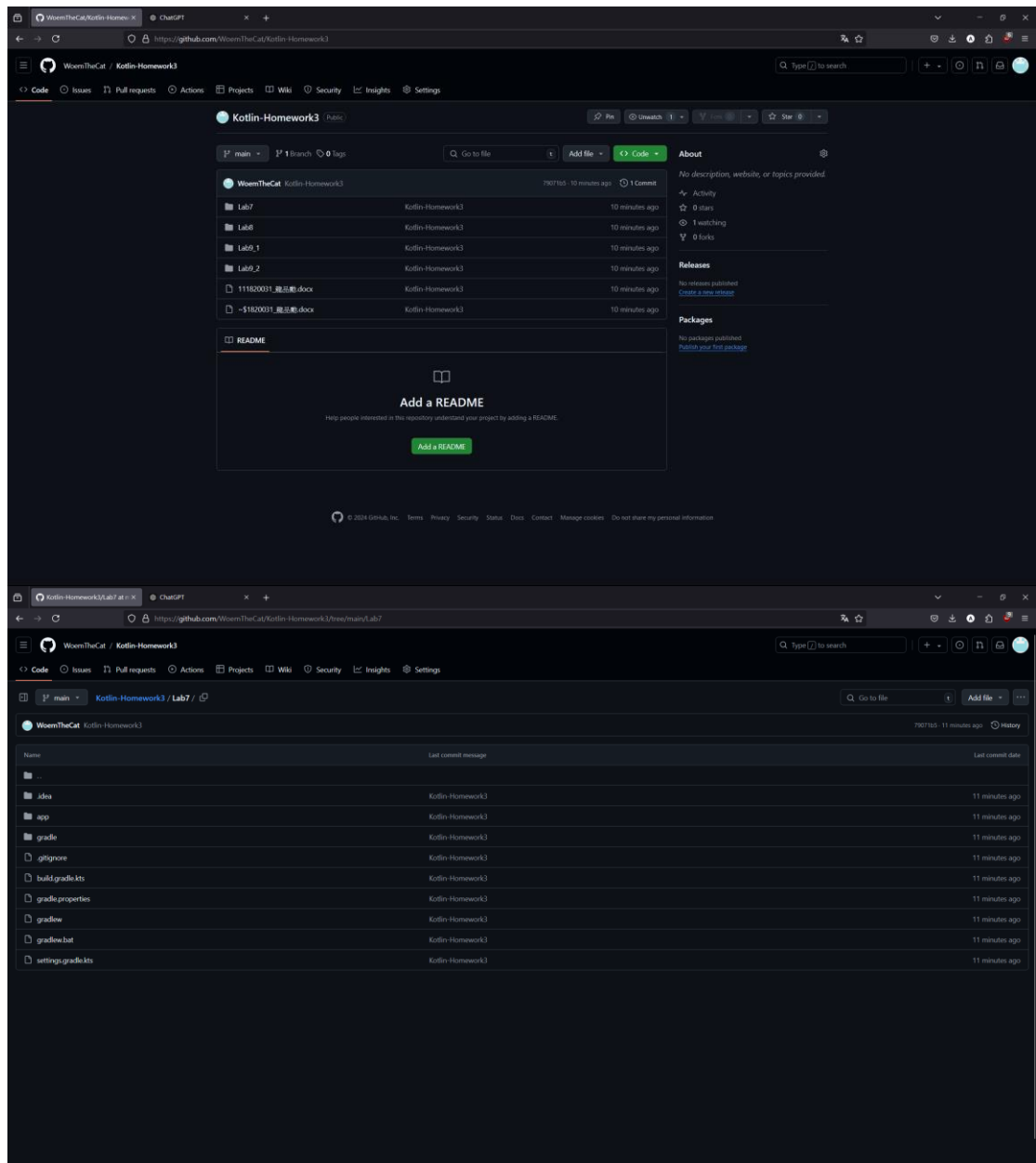
複製程式碼

```
GlobalScope.launch(Dispatchers.IO) {
    val result = performNetworkOperation()
    withContext(Dispatchers.Main) {
        textView.text = result
    }
}
```

總結

`Thread` 是處理多執行緒的基礎工具，在 Android 中主要用於執行耗時任務和處理併發操作，但開發者更推薦使用更高層的 API，例如協程或 `WorkManager`，以獲得更好的代碼可讀性和性能表現。

- GitHub 程式連結：<https://github.com/WoemTheCat/Kotlin-Homework3>
- GitHub 的截圖：



Kotlin Homework3, Lab8 at x

ChatGPT

https://github.com/WoemTheCat/Kotlin-Homework3/tree/main/Lab8

WoemTheCat / Kotlin-Homework3

Type to search

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

Kotlin-Homework3 / Lab8

Go to fileAdd file

WoemTheCat Kotlin Homework3707765 · 11 minutes agoHistory

Name	Last commit message	Last commit date
..		
idea	Kotlin Homework3	11 minutes ago
app	Kotlin Homework3	11 minutes ago
gradle	Kotlin Homework3	11 minutes ago
.gitignore	Kotlin Homework3	11 minutes ago
build.gradle.kts	Kotlin Homework3	11 minutes ago
gradle.properties	Kotlin Homework3	11 minutes ago
gradlew	Kotlin Homework3	11 minutes ago
gradlew.bat	Kotlin Homework3	11 minutes ago
settings.gradle.kts	Kotlin Homework3	11 minutes ago

Kotlin Homework3, Lab9_1 at x

ChatGPT

https://github.com/WoemTheCat/Kotlin-Homework3/tree/main/Lab9_1

WoemTheCat / Kotlin-Homework3

Type to search

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

Kotlin-Homework3 / Lab9_1

Go to fileAdd file

WoemTheCat Kotlin Homework3707765 · 11 minutes agoHistory

Name	Last commit message	Last commit date
..		
idea	Kotlin Homework3	11 minutes ago
app	Kotlin Homework3	11 minutes ago
gradle	Kotlin Homework3	11 minutes ago
.gitignore	Kotlin Homework3	11 minutes ago
build.gradle.kts	Kotlin Homework3	11 minutes ago
gradle.properties	Kotlin Homework3	11 minutes ago
gradlew	Kotlin Homework3	11 minutes ago
gradlew.bat	Kotlin Homework3	11 minutes ago
settings.gradle.kts	Kotlin Homework3	11 minutes ago

Kotlin Homework3, Lab9_2

ChatGPT

https://github.com/WisemTheCat/Kotlin-Homework3/tree/main/Lab9_2

WisemTheCat / Kotlin-Homework3

Type to search

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

main

Kotlin-Homework3 / Lab9_2

Go to file

Add file

WisemTheCat

Kotlin Homework3

7307163 · 11 minutes ago · history

name	Last commit message	Last commit date
..		
.idea	Kotlin Homework3	11 minutes ago
app	Kotlin Homework3	11 minutes ago
gradle	Kotlin Homework3	11 minutes ago
.gitignore	Kotlin Homework3	11 minutes ago
build.gradle.kts	Kotlin Homework3	11 minutes ago
gradle.properties	Kotlin Homework3	11 minutes ago
gradlew	Kotlin Homework3	11 minutes ago
gradlew.bat	Kotlin Homework3	11 minutes ago
settings.gradle.kts	Kotlin Homework3	11 minutes ago