

## 讀書會

- 組員：龍品勳、張耀文、時顯文
- 討論時間：2024/12/25 10:00 AM
- 地點：圖書館

照片：



## 心得報告

- 心得：圖書管理系統還滿實用的，但在資料有變動的時候並不會立即顯示。

- ChatGPT：

1. 請問這段 Kotlin 程式碼的意思

```
private fun setRequest() {  
    val url = "https://api.italkutalk.com/api/air"  
    val req = Request.Builder()  
        .url(url)  
        .build()  
  
    OkHttpClient().newCall(req).enqueue(object : Callback {  
        override fun onResponse(call: Call, response: Response) {  
            val json = response.body?.string()  
            val myObject = Gson().fromJson(json, MyObject::class.java)  
            showDialog(myObject)  
        }  
  
        override fun onFailure(call: Call, e: IOException) {  
            runOnUiThread {  
                btnQuery.isEnabled = true  
                Toast.makeText(this@MainActivity,  
                    "查詢失敗$e", Toast.LENGTH_SHORT  
                ).show()  
            }  
        }  
    })  
}
```



這段 Kotlin 程式碼的主要功能是使用 OkHttp 庫發送 HTTP 請求並處理回應，以下是逐步解釋：

## 1. 設定 URL

kotlin

複製程式碼

```
val url = "https://api.italkutalk.com/api/air"
```

- 定義了一個字串 `url`，表示將向該 URL 發送請求。

## 2. 建構請求

kotlin


複製程式碼

```
val req = Request.Builder()
    .url(url)
    .build()
```

- 使用 OkHttp 的 `Request.Builder()` 創建一個 `Request` 物件。
- 設定 `url` 作為目標，並通過 `build()` 方法構建請求物件。

### 3. 發送請求並處理回應

kotlin


 複製程式碼

```
OkHttpClient().newCall(req).enqueue(object : Callback {  
    override fun onResponse(call: Call, response: Response) {  
        ...  
    }  
  
    override fun onFailure(call: Call, e: IOException) {  
        ...  
    }  
})
```

- 創建一個 `OkHttpClient` 實例，並用它發送異步請求。
- `enqueue` 方法接收一個回呼函數物件 ( `Callback` )，用來非同步處理回應或錯誤。

### 4. 成功回應處理

kotlin

 複製程式碼

```
override fun onResponse(call: Call, response: Response) {  
    val json = response.body?.string()  
    val myObject = Gson().fromJson(json, MyObject::class.java)  
    showDialog(myObject)  
}
```

- `onResponse`：當伺服器成功回應時執行。
  1. 取得回應的 `body` 並將其轉換為字串（JSON 格式）。
  2. 使用 Google 的 Gson 庫將 JSON 字串解析為 `MyObject` 類的實例。
  3. 呼叫 `showDialog` 方法，將解析後的物件顯示給用戶。

## 5. 失敗處理

kotlin

複製程式碼

```
override fun onFailure(call: Call, e: IOException) {  
    runOnUiThread {  
        btnQuery.isEnabled = true  
        Toast.makeText(this@MainActivity,  
            "查詢失敗$e", Toast.LENGTH_SHORT  
        ).show()  
    }  
}
```

- `onFailure`：當請求失敗時執行。
  1. 使用 `runOnUiThread` 確保 UI 操作在主線程執行。
  2. 啟用按鈕（可能是查詢按鈕 `btnQuery`），恢復用戶操作。
  3. 顯示錯誤訊息的提示框（`Toast`），告知用戶請求失敗。

## 關鍵技術要點

### 1. OkHttp

- 用於發送 HTTP 請求和接收回應的高效 Java/Kotlin 庫。

### 2. Gson

- 將 JSON 資料與 Kotlin/Java 類互相轉換的工具。

### 3. UI 操作

- 在非主線程進行網路操作，並利用 `runOnUiThread` 切回主線程更新 UI。

### 4. 非同步

- 請求是非同步進行，不會阻塞主線程。

## 應用場景

此程式碼適合用於需要向網路 API 發送請求並處理回應的 Android 應用。例如，查詢天氣、獲取數據等。

若有具體問題需要補充解釋，請告訴我！

## 2. 請問 `ContentResolver` 的使用方法



在 Android 中，`ContentResolver` 是用來操作應用和系統中提供的 **內容提供者 (Content Provider)** 的工具。`ContentResolver` 通過 URI 來訪問內容提供者，提供查詢、插入、更新和刪除數據的功能。

以下是 `ContentResolver` 的使用方法和範例。

### 1. 獲取 `ContentResolver`

`ContentResolver` 是 Android 系統的一部分，通常通過 `Context` 來獲取：

kotlin

複製程式碼

```
val contentResolver = context.contentResolver
```

在 Activity 中可以直接使用：

kotlin

複製程式碼

```
val contentResolver = contentResolver
```

## 2. ContentResolver 的主要操作

`ContentResolver` 提供了以下方法來操作數據：

| 方法                            | 功能                   |
|-------------------------------|----------------------|
| <code>query</code>            | 查詢數據                 |
| <code>insert</code>           | 插入數據                 |
| <code>update</code>           | 更新數據                 |
| <code>delete</code>           | 刪除數據                 |
| <code>getType</code>          | 獲取 URI 對應的 MIME 類型   |
| <code>openInputStream</code>  | 讀取 URI 指定資源的輸入流（如圖片） |
| <code>openOutputStream</code> | 寫入 URI 指定資源的輸出流      |

## 3. 常見的 URI 範例

- 內建的內容提供者 URI

- 聯絡人數據：`ContactsContract.Contacts.CONTENT_URI`
- 短信數據：`Telephony.Sms.CONTENT_URI`
- 多媒體數據（圖片、音樂、影片）：`MediaStore.Images.Media.EXTERNAL_CONTENT_URI`

- 自定義內容提供者 URI


- 由開發者自行定義，例如 `content://com.example.app.provider/table_name`

## 4. 使用範例

### (1) 查詢數據

從系統的聯絡人數據庫查詢聯絡人：

kotlin

 複製程式碼


```
val uri = ContactsContract.Contacts.CONTENT_URI
val projection = arrayOf(ContactsContract.Contacts.DISPLAY_NAME)
val cursor = contentResolver.query(uri, projection, null, null, null)

cursor?.use {
    while (it.moveToNext()) {
        val name = it.getString(it.getColumnIndexOrThrow(ContactsContract.Contacts.DISPLAY_NAME))
        println("聯絡人姓名: $name")
    }
}
```

### (2) 插入數據

向自定義內容提供者插入數據：

kotlin

 複製程式碼

```
val uri = Uri.parse("content://com.example.app.provider/table_name")
val values = ContentValues().apply {
    put("name", "John Doe")
    put("age", 30)
}

val newUri = contentResolver.insert(uri, values)
println("插入的數據 URI: $newUri")
```





### (3) 更新數據

更新特定條件的數據：

kotlin

複製程式碼

```
val uri = Uri.parse("content://com.example.app.provider/table_name")
val values = ContentValues().apply {
    put("age", 31)
}
val selection = "name = ?"
val selectionArgs = arrayOf("John Doe")

val rowsUpdated = contentResolver.update(uri, values, selection, selectionArgs)
println("更新的行數: $rowsUpdated")
```

### (4) 刪除數據

刪除特定條件的數據：

kotlin

複製程式碼

```
val uri = Uri.parse("content://com.example.app.provider/table_name")
val selection = "age < ?"
val selectionArgs = arrayOf("18")

val rowsDeleted = contentResolver.delete(uri, selection, selectionArgs)
println("刪除的行數: $rowsDeleted")
```

## (5) 操作多媒體資源

讀取系統圖片的輸入流：

kotlin

複製程式碼

```
val uri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI
val projection = arrayOf(MediaStore.Images.Media._ID, MediaStore.Images.Media.DISPLAY_NAME)

val cursor = contentResolver.query(uri, projection, null, null, null)
cursor?.use {
    while (it.moveToNext()) {
        val id = it.getLong(it.getColumnIndexOrThrow(MediaStore.Images.Media._ID))
        val displayName = it.getString(it.getColumnIndexOrThrow(MediaStore.Images.Media.DISPLAY_NAME))

        val imageUri = ContentUris.withAppendedId(uri, id)
        println("圖片名稱: $displayName, URI: $imageUri")
    }
}
```

## 5. 注意事項

### 1. 權限

- 使用某些內容提供者時需要申請權限，例如訪問聯絡人或圖片需要以下權限：

xml

複製程式碼

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

### 2. 版本差異

- 某些系統內容提供者的行為會隨 Android 版本變化。例如 Android 10 及以後的版本需要處理 Scoped Storage。

### 3. 效率

- 查詢和操作大量數據時，注意設置合適的條件和範圍，避免性能問題。

若你有具體的應用場景或問題，歡迎提供詳細需求，我可以給出更精準的範例！

## 3. 請問在 http 中 get 和 post 的差異



在 HTTP 協議中，**GET** 和 **POST** 是最常見的兩種請求方法，它們主要用於客戶端（如瀏覽器）與伺服器之間的數據傳輸。以下是它們的主要區別：

## 1. GET 方法

GET 用於從伺服器獲取資源或數據。

### 特點

- 用於查詢或讀取數據
  - 適合不會修改伺服器狀態的操作，例如取得資料列表或檢視單個項目。
- 參數放在 URL 中
  - 請求的參數附加在 URL 後方，格式為 `?key1=value1&key2=value2`。
  - 示例：

```
http
```

[複製程式碼](#)

```
GET /search?q=example HTTP/1.1
Host: www.example.com
```

- 無需請求主體
  - GET 請求一般只包括 URL 和標頭，不包含請求主體。

### 優點

1. 簡單直觀：直接在地址欄輸入 URL 即可發送請求。
2. 可被緩存：瀏覽器和伺服器可以緩存 GET 請求的響應。
3. 可被書籤保存：因為參數在 URL 中，可方便保存或分享。
4. 可被搜索引擎索引：有助於 SEO。



### 缺點

1. **參數長度限制**：URL 的長度受到瀏覽器或伺服器的限制（通常是 2048 字符）。
2. **安全性低**：參數直接暴露在 URL 中，容易被竊取或篡改。

## 2. POST 方法

POST 用於向伺服器提交數據或進行操作。

### 特點

- **用於傳輸數據或進行操作**
  - 適合會改變伺服器狀態的操作，例如新增、修改或刪除資源。
- **參數放在請求主體中**
  - 參數不會暴露在 URL 中，而是包含在請求主體內。
- 示例：

http

複製程式碼

```
POST /api/login HTTP/1.1
Host: www.example.com
Content-Type: application/json

{
  "username": "user123",
  "password": "pass456"
}
```

- **支持更大的數據量**
  - 由於參數在主體內，POST 請求理論上不受長度限制。

優點

- 1. 安全性更高：參數不直接暴露在 URL 中，不易被竊取。
- 2. 無長度限制：適合傳輸大數據或文件。
- 3. 支持複雜數據：可以傳輸 JSON、XML 或文件等格式。

缺點

- 1. 不可緩存：POST 請求通常不會被瀏覽器或伺服器緩存。
- 2. 不可被書籤保存：參數不在 URL 中，無法直接保存。
- 3. 相對耗資源：處理 POST 請求可能比 GET 請求需要更多伺服器資源。

3. GET 和 POST 的對比表

| 特性       | GET                 | POST                |
|----------|---------------------|---------------------|
| 目的       | 獲取資源或數據             | 提交數據或改變伺服器狀態        |
| 參數位置     | URL ( 查詢字串 )        | 請求主體                |
| 數據大小限制   | 有限 ( 取決於瀏覽器或伺服器限制 ) | 理論上無限制 ( 受伺服器配置影響 ) |
| 安全性      | 參數暴露在 URL 中，安全性較低   | 參數隱藏在請求主體中，相對安全     |
| 是否可緩存    | 可以被浏览器和伺服器緩存        | 一般不可被緩存             |
| 是否可被書籤保存 | 可以                  | 不可以                 |
| 是否適合敏感數據 | 不適合，參數容易暴露          | 適合                  |
| 用例       | 查詢資源、檢索數據           | 提交表單、上傳文件、修改數據      |

## 4. 使用場景

- GET
  - 搜索引擎查詢
  - 獲取天氣信息
  - 查看產品列表
  - 請求靜態資源 ( 圖片、JS、CSS )
- POST
  - 用戶註冊、登錄
  - 文件上傳
  - 提交表單
  - 創建或更新數據

## 5. 注意事項

### 1. 安全性

- POST 比 GET 更安全，但仍需通過 HTTPS 保護數據傳輸。
- 不要將敏感數據 ( 如密碼、Token ) 包含在 URL 中。

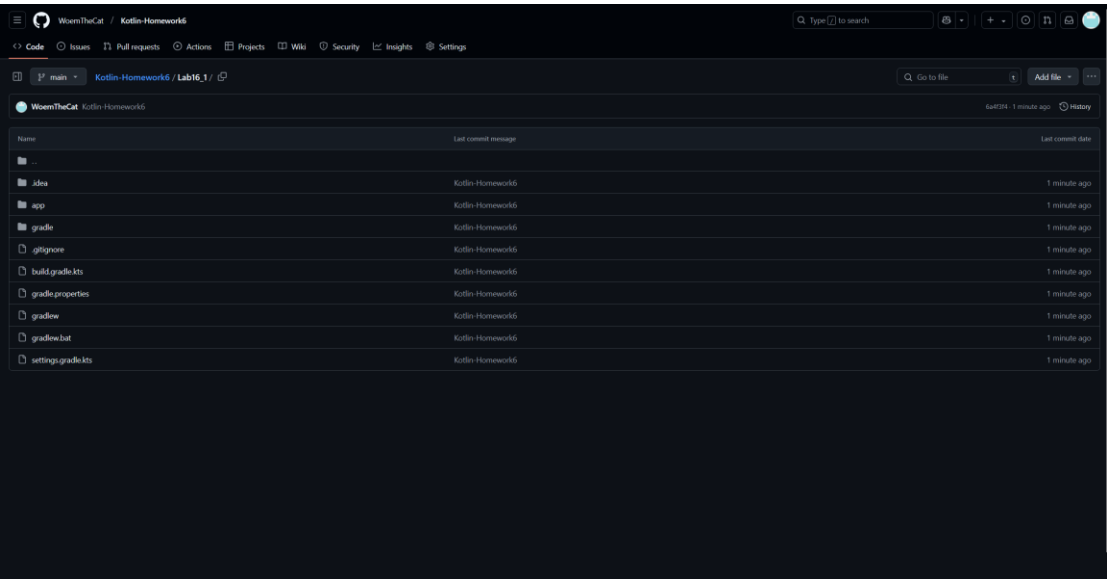
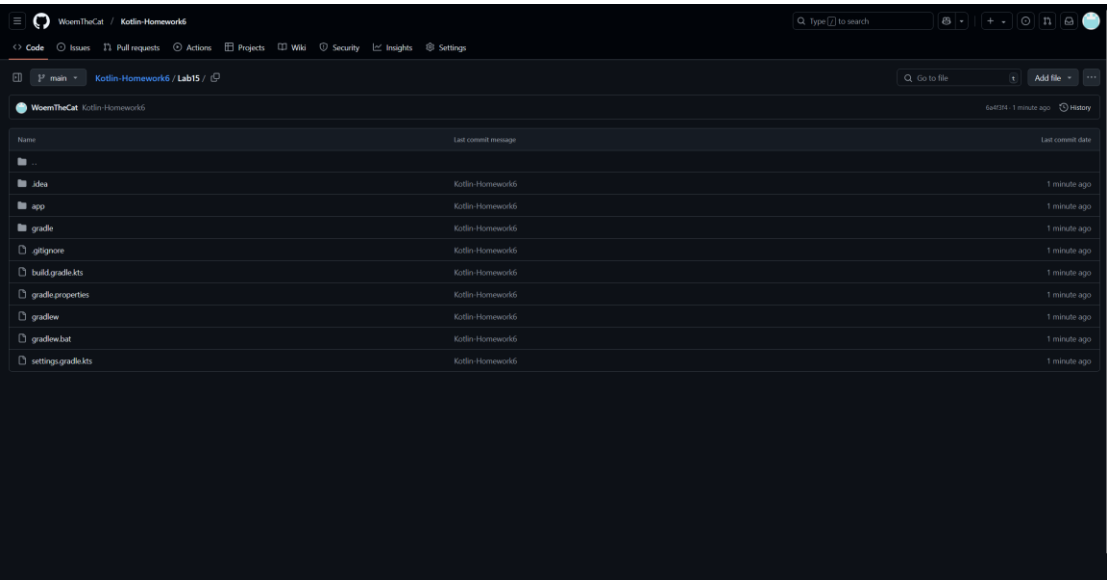
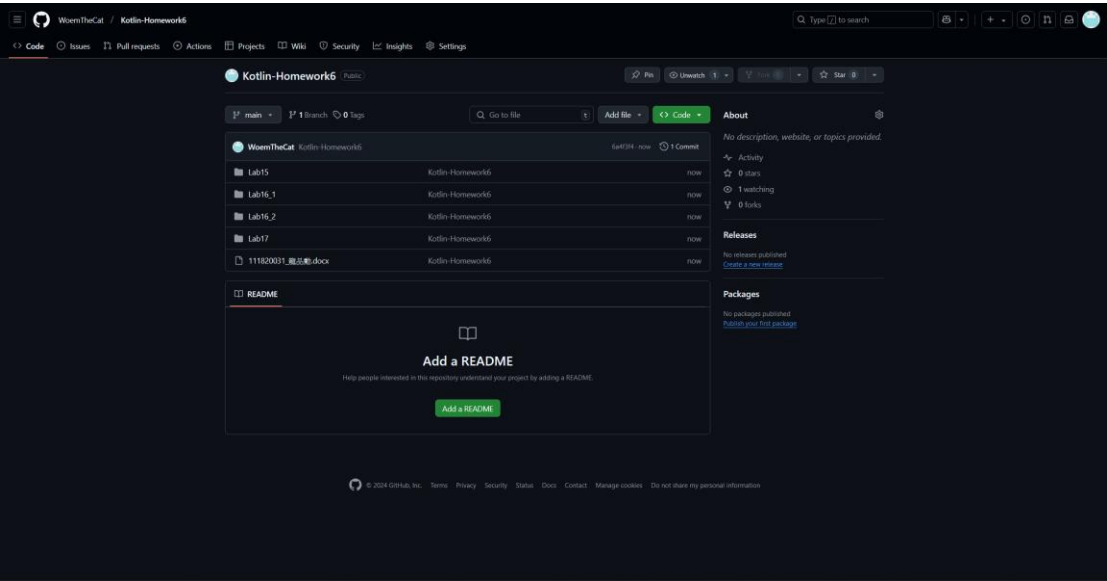
### 2. 效能

- GET 請求相對輕量，應用於不需要提交大量數據的場景。
- POST 適合需要更複雜數據處理的場景，但可能消耗更多資源。

若有進一步的問題，歡迎提出！



- GitHub 程式連結：<https://github.com/WoemTheCat/Kotlin-Homework6>
- GitHub 的截圖：



WiornTheCat / Kotlin-Homework6

Type to search

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

Kotlin-Homework6 / Lab16.2

Go to file

Add file

...

WiornTheCat Kotlin-Homework6

6a7394 1 minute ago

History

| Name                | Last commit message | Last commit date |
|---------------------|---------------------|------------------|
| ..                  |                     |                  |
| idea                | Kotlin-Homework6    | 1 minute ago     |
| app                 | Kotlin-Homework6    | 1 minute ago     |
| gradle              | Kotlin-Homework6    | 1 minute ago     |
| .gitignore          | Kotlin-Homework6    | 1 minute ago     |
| build.gradle.kts    | Kotlin-Homework6    | 1 minute ago     |
| gradle.properties   | Kotlin-Homework6    | 1 minute ago     |
| gradlew             | Kotlin-Homework6    | 1 minute ago     |
| gradlew.bat         | Kotlin-Homework6    | 1 minute ago     |
| settings.gradle.kts | Kotlin-Homework6    | 1 minute ago     |

WiornTheCat / Kotlin-Homework6

Type to search

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

Kotlin-Homework6 / Lab17

Go to file

Add file

...

WiornTheCat Kotlin-Homework6

6a7394 2 minutes ago

History

| Name                | Last commit message | Last commit date |
|---------------------|---------------------|------------------|
| ..                  |                     |                  |
| idea                | Kotlin-Homework6    | 2 minutes ago    |
| app                 | Kotlin-Homework6    | 2 minutes ago    |
| gradle              | Kotlin-Homework6    | 2 minutes ago    |
| .gitignore          | Kotlin-Homework6    | 2 minutes ago    |
| build.gradle.kts    | Kotlin-Homework6    | 2 minutes ago    |
| gradle.properties   | Kotlin-Homework6    | 2 minutes ago    |
| gradlew             | Kotlin-Homework6    | 2 minutes ago    |
| gradlew.bat         | Kotlin-Homework6    | 2 minutes ago    |
| settings.gradle.kts | Kotlin-Homework6    | 2 minutes ago    |