

讀書會

- 組員：龍品勳、張耀文、時顯文
- 討論時間：2024/12/4 10:00 AM
- 地點：圖書館

照片：



心得報告

- 心得：廣播的操作比起之前的作業還要複雜許多。
- ChatGPT：

1. 請問 START_NOT_STICKY 在 android studio 中的意思



在 Android 開發中，`START_NOT_STICKY` 是一種服務 (Service) 的啟動模式，指定了當服務被系統終止後的行為。它是 `onStartCommand()` 方法返回值中的一種選項。

定義

當你的服務在執行中被系統強制終止（例如因為內存不足），系統不會自動重新啟動服務，除非明確地再次啟動它。

使用場景

適合用於那些 不需要持續運行，且用戶主動交互後即可完成的任務。

例如：

- 你有一個簡單的服務用來處理一次性任務（如下載一個小文件）。
- 如果服務在完成前被系統中止，你可以允許用戶自行重新啟動。

回傳值的完整選項

在 `onStartCommand(Intent intent, int flags, int startId)` 中，你可以返回以下值之一：

值	行為
<code>START_NOT_STICKY</code>	如果服務被終止， 不要重新啟動 ，除非用戶或其他代碼明確重新啟動它。
<code>START_STICKY</code>	如果服務被終止，系統會嘗試重新啟動服務，但不會傳遞原始的 Intent。主要用於持續運行的服務（例如音樂播放器）。
<code>START_REDELIVER_INTENT</code>	如果服務被終止，系統會重新啟動並傳遞原來的 Intent。用於需要重新執行任務的場景。
<code>START_STICKY_COMPATIBILITY</code>	與 <code>START_STICKY</code> 相似，但保證與更舊版本的 Android 系統兼容。

實際範例

java

複製程式碼

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    // 執行服務邏輯
    Log.d("MyService", "Service is running");

    // 指定啟動模式
    return START_NOT_STICKY;
}
```

總結

`START_NOT_STICKY` 主要用於不需要持續運行或保存狀態的輕量級任務，適合那些在被終止後可由用戶或應用邏輯重新啟動的服務。

2. 請問這段程式碼的意思

```
private val receiver =
    object : BroadcastReceiver() {
        override fun onReceive(context: Context, intent: Intent) {
            intent.extras?.let {
                val tvMsg = findViewById<TextView>(R.id.tvMsg)
                tvMsg.text = "${it.getString("msg")}"
            }
        }
    }
```



這段程式碼定義了一個 **廣播接收器 (BroadcastReceiver)**，用來處理從系統或其他應用程式發送的廣播。讓我們逐步解釋這段程式碼的意思：

程式碼的結構與含義

kotlin

複製程式碼

```
private val receiver =
    object : BroadcastReceiver() {
        override fun onReceive(context: Context, intent: Intent) {
            intent.extras?.let {
                val tvMsg = findViewById<TextView>(R.id.tvMsg)
                tvMsg.text = "${it.getString("msg")}"
            }
        }
    }
}
```

1. 建立一個匿名的 `BroadcastReceiver` 實例

kotlin

複製程式碼

```
private val receiver = object : BroadcastReceiver() {
    // ...
}
```

- `private val receiver`: 定義了一個廣播接收器的物件，名為 `receiver`，它是一個類別的實例。
- `object : BroadcastReceiver()`: 使用匿名類別的方式實現 `BroadcastReceiver`，覆寫了它的抽象方法 `onReceive()`。

2. 覆寫 `onReceive` 方法

kotlin

複製程式碼

```
override fun onReceive(context: Context, intent: Intent) {  
    // 處理廣播訊息的邏輯  
}
```

- `onReceive(context: Context, intent: Intent)`:
 - `context`: 當前廣播接收器的上下文 (Context)，通常是應用程式的環境。
 - `intent`: 廣播中攜帶的數據和額外訊息，類似於一個信封，內含需要處理的內容。

3. 讀取廣播攜帶的數據

kotlin

複製程式碼

```
intent.extras?.let {  
    val tvMsg = findViewById<TextView>(R.id.tvMsg)  
    tvMsg.text = "${it.getString("msg")}"  
}
```

- `intent.extras`：從廣播的 `intent` 中提取附加數據 (`Bundle`) 。
 - 如果 `extras` 不為空，則進入 `let` 區塊執行。
- `it.getString("msg")`：從附加數據中提取名為 `"msg"` 的字串值。
- `findViewById<TextView>(R.id.tvMsg)`：
 - 找到佈局中的 `TextView` 控件 (假設其 ID 為 `tvMsg`) 。
- `tvMsg.text`：將提取到的訊息顯示在 `TextView` 中。


用途

- 這段程式碼的作用是：
 1. 接收一個廣播。
 2. 從廣播的 `intent` 中讀取一個名為 `"msg"` 的字串訊息。
 3. 將這個訊息顯示在界面上的 `TextView` 控件中。

範例使用場景

假設有另一個地方發送了一個廣播，帶有一個 `"msg"` 的訊息，像這樣：

kotlin

 複製程式碼

```
val intent = Intent("com.example.CUSTOM_BROADCAST")
intent.putExtra("msg", "Hello, Broadcast!")
sendBroadcast(intent)
```

這時，`receiver` 會觸發 `onReceive()`，從廣播中提取 `"Hello, Broadcast!"` 並將其顯示在 `TextView` 中。

注意事項

1. 廣播接收器的註冊：必須使用動態或靜態方式註冊這個接收器，例如：

kotlin

複製程式碼

```
registerReceiver(receiver, IntentFilter("com.example.CUSTOM_BROADCAST"))
```

2. 動態註冊需記得解除註冊：防止內存洩漏。

kotlin

複製程式碼

```
unregisterReceiver(receiver)
```

3. 請問這段程式碼的意思

```
override fun onDestroy() {  
    super.onDestroy()  
    unregisterReceiver(receiver)  
}
```



這段程式碼是用來處理 清理廣播接收器 (`BroadcastReceiver`) 的邏輯，確保應用在結束時釋放資源並避免潛在的 內存洩漏 或 意外行為。

程式碼解析

kotlin

複製程式碼

```
override fun onDestroy() {  
    super.onDestroy()  
    unregisterReceiver(receiver)  
}
```

1. 覆寫 `onDestroy()` 方法

- `onDestroy()` 是 Android 中的一個生命週期方法，當活動 (Activity) 或服務 (Service) 被銷毀時會被呼叫。
- `super.onDestroy()` :
 - 呼叫父類的 `onDestroy()` 方法，確保系統處理了基礎的銷毀邏輯。
 - 例如：釋放系統資源、停止任何正在運行的操作。

2. 解除廣播接收器

kotlin

複製程式碼

```
unregisterReceiver(receiver)
```

- `unregisterReceiver(receiver)`：
 - 告訴系統不再需要這個 `receiver`，解除它的註冊。
 - 這樣做可以防止活動或服務被銷毀後，廣播仍然試圖觸發已經不存在的 `receiver`，從而導致錯誤。

作用

- 釋放資源：確保 `receiver` 不再監聽廣播。
- 防止內存洩漏：如果未解除註冊，系統會保留 `receiver` 的引用，導致活動或服務無法正確回收。
- 避免異常：若未解除註冊，當廣播到達而 `receiver` 不存在時，應用可能會崩潰。

使用場景

通常在以下情況下，需要在 `onDestroy()` 中解除廣播接收器：

1. 動態註冊的廣播接收器：通過 `registerReceiver()` 註冊的接收器，必須在活動或服務的生命週期結束時手動解除。

kotlin

複製程式碼

```
registerReceiver(receiver, IntentFilter("com.example.CUSTOM_BROADCAST"))
```

2. 活動 (Activity) 或 服務 (Service) 被銷毀時：確保不再處理任何廣播訊息。

總結

這段程式碼的目的是在活動或服務被銷毀時，正確地解除廣播接收器的註冊，釋放資源，並保證應用的穩定性與效能。

- [GitHub 程式連結](#)：
- [GitHub 的截圖](#)：