Module VI
Android Components

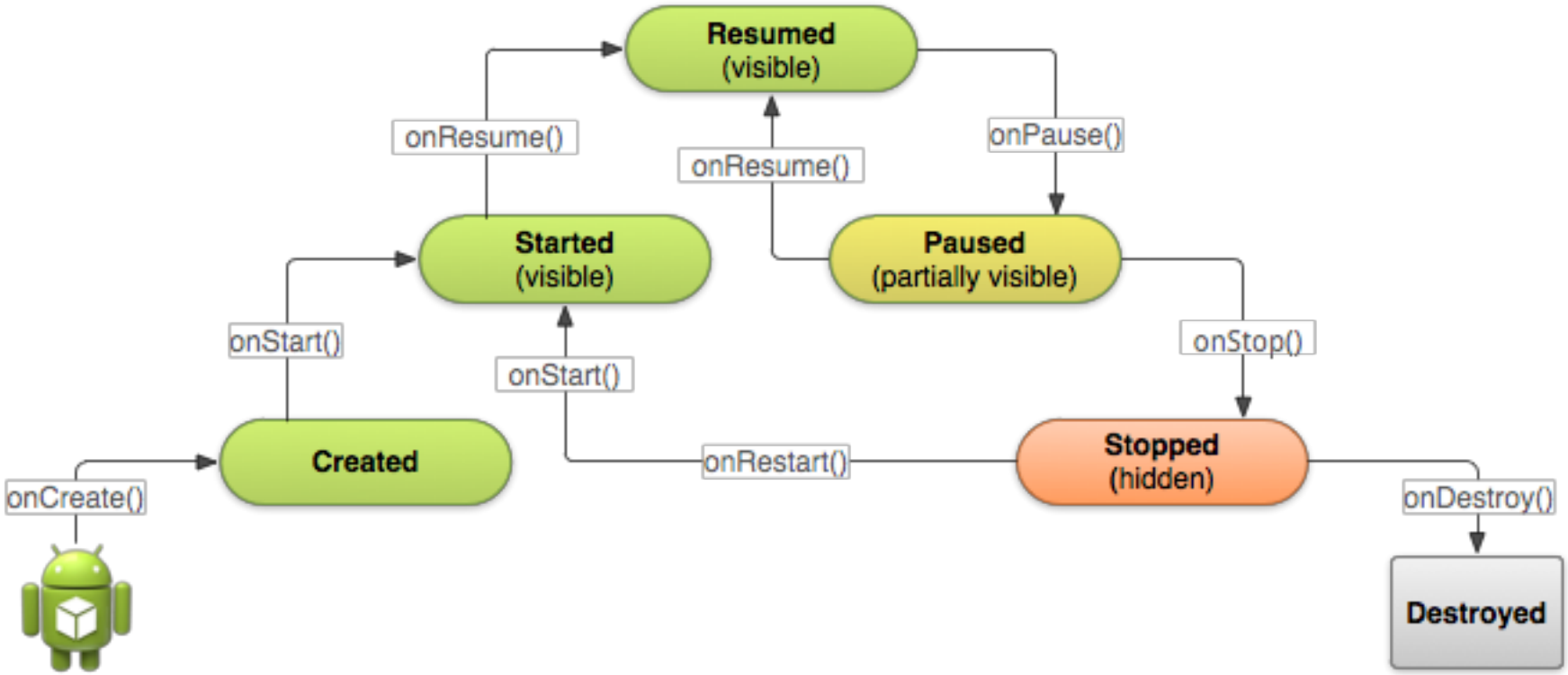
- Activity
- Intent
- Notification
- Broadcast
- Service

Activity生命週期

- Activity主要功能就是建立一個可以讓使用者對行動裝置互動的畫面

- Activity從開始執行到結束，會經歷七個階段，並自動呼叫對應以下七種方法：
  - onCreate()　：Activity第一次被建立時呼叫
  - onStart()　：Activity畫面要呈現時呼叫
  - onResume()　：將要與使用者進行互動時呼叫
  - onPause()　：畫面要被其它Activity取代時呼叫
  - onStop()　：畫面被取代時呼叫
  - onRestart()　：回復到onStart時呼叫
  - onDestroy()　：Activity準備結束時呼叫

Activity生命週期示意圖



範例：ActivityLifeCycleEx

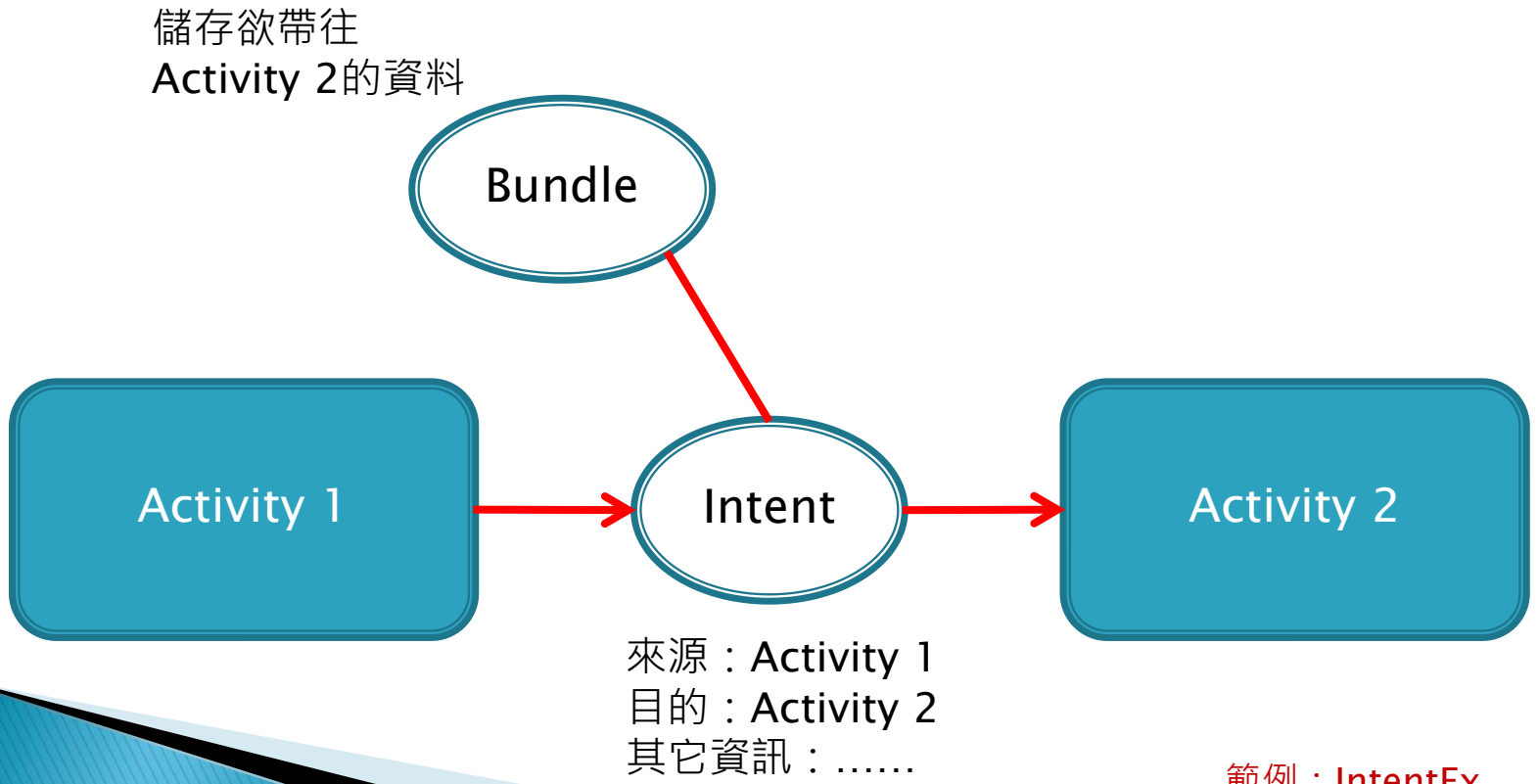
行程(Process)

- 行程是由Android系統掌控，當記憶體不足時，可能會隨時終止某一個Activity行程。不重要的行程容易被終止。從重要到不重要的行程排列如下：

  - 前景行程：onCreate()、onStart()與onResume()
  - 可視行程：非前景行程，而使用者能夠看到畫面，即onPause()
  - 服務行程：該行程啟動後會保持執行狀態，即Service
  - 背景行程：畫面被完全切換，即onStop()
  - 空行程　：背景行程被終止會進入到空行程，但Activity仍會存在，以便快速恢復到前景行程

Activity間傳遞資料

- 在兩個Activity作切換時，可以附帶額外資料，只要將該資料儲存在Bundle，而讓Bundle依附在Intent上即可



範例：IntentEx

Activity回傳資料

- 讓被呼叫的Activity回傳資料也是使用Intent物件。跟前面不同的是，不是呼叫startActivity()，而是改為呼叫startActivityForResult()

- 同樣利用Intent物件傳遞資料，但需要多附帶一個請求代碼，用來確認回傳資料的來源

- 回傳給原Activity時，系統會將結果代碼與請求代碼一起傳給onActivityResult()方法，預設結果代碼：
  - RESULT_OK：回傳成功
  - RESULT_CANCELED：回傳取消

範例：ActivityReturnEx

Intent Filter

- Intent物件除了上述功能，如切換activity、傳遞資料與取得回傳資料外，還能夠發出求助訊號，找到適合開啟的應用程式來處理

- 在AndroidManiFest.xml檔裡可以設定intent-filter資訊，讓系統可以挑選合適的相關程式

- intent-filter相關標籤說明：
  - <action.../>：描述這個activity可執行的操作類型
    - 如：VIEW、EDIT、DIAL、SEND等
  - <data.../>：描述這個activity可處理的資料或資料型態
    - 如：image/*、video/*等

    (android:scheme也可以指定類型，如http、tel、file等)
  - <category.../>：指定這個activity類型
    - 除了開啟預設LAUNCHER外，其它通常設為DEFAULT

範例：IntentFilterEx

Notification (1/2)

- 欲實現通知訊息顯示在狀態列上，步驟如下：
  - 呼叫Activity.getSystemService()得到NotificationManager物件
  - 呼叫Notification.Builder建構子建立Notification.Builder物件後，再呼叫build()方法即可得到Notification物件
  - 設定標題、圖示與內容相關資訊
  - 呼叫NotificationManager.notify()發送通知訊息
  - 呼叫NotificationManager.cancel()刪除通知訊息；若呼叫setAutoCancel()則會自動將狀態列上的訊息移除
  - 也可以指定當使用者點選訊息後，會前往的Activity

Notification (2/2)

- Notification.Builder必須在API level 11以上的裝置才能使用，但在API level 16以後，Notification.Builder改為使用build()方法建立，所以需要將專案的Android:minSdkVersion屬性設定為16以上
- 若是要讓專案也能支援Android 2.X，需要換成support-v4 library裡的NotificationCompat.Builder，兩者用法非常類似
- 有關API level設定，在Android Studio已移到Gradle script檔案裡(Module: app)，而不再是AndroidManifest.xml檔案
- Gradle script可處理API level調整、專案版本管理與添加外部函式庫等相關設定

範例：NotificationEx

Broadcast Receiver (1/3)

- 可以在AndroidManifest.xml設定要攔截的Broadcast並做相關對應的處理，步驟如下：
  - 繼承BroadcastReceiver類別，並改寫onReceive()方法
  - 設定要攔截何種action訊息的Broadcast
  - 發出Broadcast被攔截時，就會自動執行onReceive()方法
  - 要攔截簡訊或來電等，需要再額外做uses-permission設定

- uses-permission：
  - 應用程式有對使用者產生重大影響的功能時，需在AndroidManifest.xml檔裡設定<uses-permission>，讓使用者在安裝應用程式前產生對應的告知訊息，讓使用者決定是否要安裝
  - 相關permission參考：

    http://developer.android.com/reference/android/Manifest.permission.html

Broadcast Receiver (2/3)

- 攔截簡訊action：
  · android.provider.Telephony.SMS_RECEIVED

- 設定permission：
  · android.permission.RECEIVE_SMS

- 簡訊為PDU(Protocol Description Unit)格式

- 取得bundle內儲存的PDU資料
  · Object[ ] pdus = (Object[ ])bundle.get(“pdus”);

- SmsMessage相關方法：
  · createFromPdu()：將PDU資料轉成SmsMessage物件
  · getDisplayMessageBody()：取得簡訊內容
  · getDisplayOriginatingAddress()：取得發訊位址
  · getTimestampMillis()：取得訊息時間

Broadcast Receiver (3/3)

- 攔截來電action：
  - android.intent.action.PHONE_STATE
- 設定permission：
  - android.permission.READ_PHONE_STATE
- 來電中可取得手機狀態與來電號碼
- 取得手機狀態：
  - bundle.getString(TelephonyManager.EXTRA_STATE);
- 來電中可取得手機號碼：
  - Bundle.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);

- 註：Android 6.0以後，上述兩項權限需在執行階段取得使用者同意
- (參考範例說明)

範例：BroadcastReceiverEx

Broadcast Sender

- 自行發送Broadcast後也可再攔截此Broadcast：
  - 繼承BroadcastReceiver類別，並改寫onReceive()方法
  - 呼叫Context的registerReceiver()註冊BroadcastReceiver
  - 呼叫Context的sendBroadcast()方法發出Broadcast
  - 系統自動呼叫onReceive()方法
  - 呼叫Context的unregisterReceiver()方法解除BroadcastReceiver的註冊

範例：BroadcastSenderEx

Service LifeCycle (1/2)

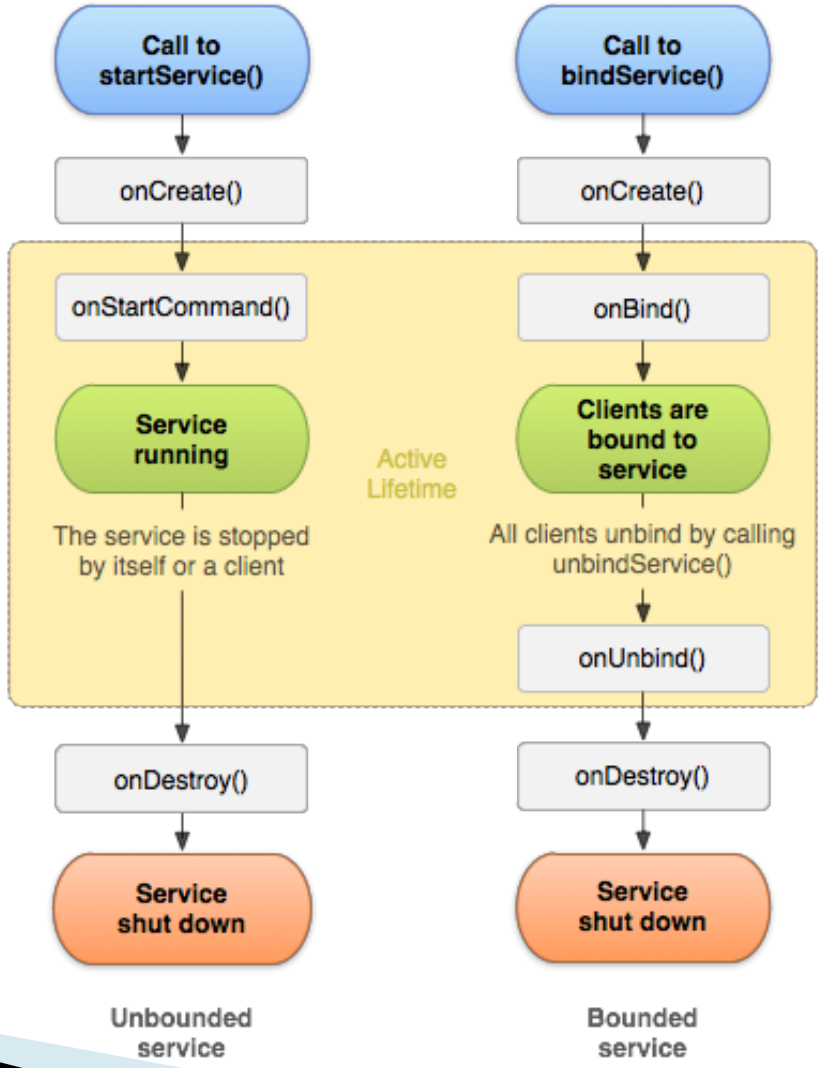
- Service與Activity都是Context(又稱環境或是容器)，差別在於Activity有UI畫面可以跟使用者互動，而Service沒有UI畫面，故無法與使用者進行互動，屬於服務行程在背景長時間執行著

- Service必須在AndroidManifest.xml檔裡使用<service>設定

- 有兩種方式可以啟動Service：
  - 呼叫Context的startService()開啟Service
  - 呼叫Context的bindService()連結Service，若Service未開啟就會在呼叫後開啟Serivce

Service LifeCycle (2/2)

- 呼叫startService()開啟Service
- 系統呼叫onCreate()
- 系統呼叫onStartCommand()
- 若呼叫stopService()，則系統會自動呼叫onDestroy()關閉Service



- 呼叫bindService()會連結Service，若Service尚未開啟，就會在此時建立
- 系統呼叫onCreate()，但不會呼叫onStartCommand()
- 系統呼叫onBind()並回傳IBinder物件
- 將IBinder物件當做參數傳遞給onServiceConnected()

範例： StartServiceEx
ServiceBindEx

IntentService

- IntentService是Service的子類別，會自動開啟一個新的工作執行序，並在執行完畢時自動關閉
- 要使用IntentService功能必須先繼承它，然後改寫onHandleIntent (Intent intent)方法；當有請求時會自動執行onHandleIntent()
- 範例為攔截開機成功後，即啟動IntentService並開啟一個Activity
- 攔截開機成功的action
  - android.intent.action.BOOT_COMPLETED
- 設定<uses-permission>
  - android.permission.RECEIVE_BOOT_COMPLETED

範例：IntentServiceEx