第十七章

SERVICE

SERVICE簡介

SERVICE簡介

- Android的Service提供開發者機制製作背景服務
- Service不用搭配Activity，也就是不用前景畫面即可執行
- 時常搭配的Broadcast Receiver或Activity
  - 例如收到開機通知時，啟動蒐集使用者資訊的背景程式
  - 例如App選擇上傳照片後，由上傳照片服務負責上傳，使用者就可以開啟別的App

SERVICE簡介

- Android的Service分成兩種
- 一般背景執行的背景服務
- 與App綁在一起的背景服務

- 一般背景執行的服務是本章介紹的重點

Service/StartServiceBasic

註冊服務

註冊服務

- Service屬於Android四大基本載體之一，所以使用前要在AndroidManifest.xml中宣告

- 本例子如下
  - EditText打字，把文字送到Service，由Service顯示文字在Notification，藉此來了解Service的運作與使用方式

註冊服務

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
 <application ... >
   <service
      android:name=".SimpleService"
      android:exported="true">
      <intent-filter>
          <action android:name="com.example.intent.START_SERVICE"/>
      </intent-filter>
   </service>
 </application>
</manifest>
```

- Service的註冊寫在<application>之中

註冊服務

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
 <application ... >
   <service
      android:name=".SimpleService"
      android:exported="true">
      <intent-filter>
          <action android:name="com.example.intent.START_SERVICE"/>
      </intent-filter>
   </service>
 </application>
</manifest>
```

- android:name填寫Service的Class名稱
- android:exported設定為true表示Service可以接受app以外的Intent來啟動

註冊服務

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
 <application ... >
   <service
      android:name=".SimpleService"
      android:exported="true">
      <intent-filter>
          <action android:name="com.example.intent.START_SERVICE"/>
      </intent-filter>
   </service>
 </application>
</manifest>
```

- 註冊intent-filter，標註可以啟動Service的Intent Action是什麼

Service/StartServiceBasic

建立SERVICE

建立SERVICE

- 建立一般的Service時，應該是Class繼承Service來製作，但Android官方並不建議這麼做，因為Service的開發若是沒做好，很容易影響到整體手˘機的效能
- Android推薦一般的Service使用IntentService

建立SERVICE

```
public class SimpleService extends IntentService {
	public SimpleService() {
		super("SimpleService");
	}
	@Override
	public void onCreate() {
		super.onCreate();
	}
	@Override
	protected void onHandleIntent(Intent intent) {
		...
	}
	@Override
	public void onDestroy() {
		super.onDestroy();
	}
}
```

建立SERVICE

```
public class SimpleService extends IntentService {
      public SimpleService() {
            super("SimpleService");
      }
      ……
}
```

- Class要繼承IntentService
- 一定要建立一個空白(預設)建構子

建立SERVICE

```
public class SimpleService extends IntentService {
        ……
        public void onCreate() {
                super.onCreate();
        }
        ……
        public void onDestroy() {
                super.onDestroy();
        }
}
```

- Service也有生命週期，當Service被啟動時，onCreate()就會被系統呼叫
- 當Service結束時，onDestroy()會被系統呼叫

建立SERVICE

```
public class SimpleService extends IntentService {
        ……
        @Override
        protected void onHandleIntent(Intent intent) {
                ...
        }
        …
}
```

- IntentService可以從onHandleIntent()收到Intent
- 處理方式與activity或receiver處理Intent的方式相同

建立SERVICE

- IntentService的特點就是在開發者只要專注在onHandleIntent()之中即可

- IntentService的另一個特點是，當onHandleIntent()內的工作執行完畢，Service會自動停止(onDestroy)

Service/StartServiceBasic

啟動SERVICE

啟動SERVICE

```
Intent intent = new
Intent("com.example.intent.START_SERVICE");
intent.putExtra("text", message);
startService(intent);
```

- 首先建立一個Intent，使用的Action是Service在AndroidManifest.xml中註冊時，intent-filter內註冊的Intent Action

- 使用intent.putExtra()放入要帶到Service的數值

啟動SERVICE

```
Intent intent = new
Intent("com.example.intent.START_SERVICE");
intent.putExtra("text", message);
startService(intent);
```

- 呼叫startService(Intent)啟動服務
  - Android系統會依照Intent設定的條件啟動對應的Service

停止SERVICE

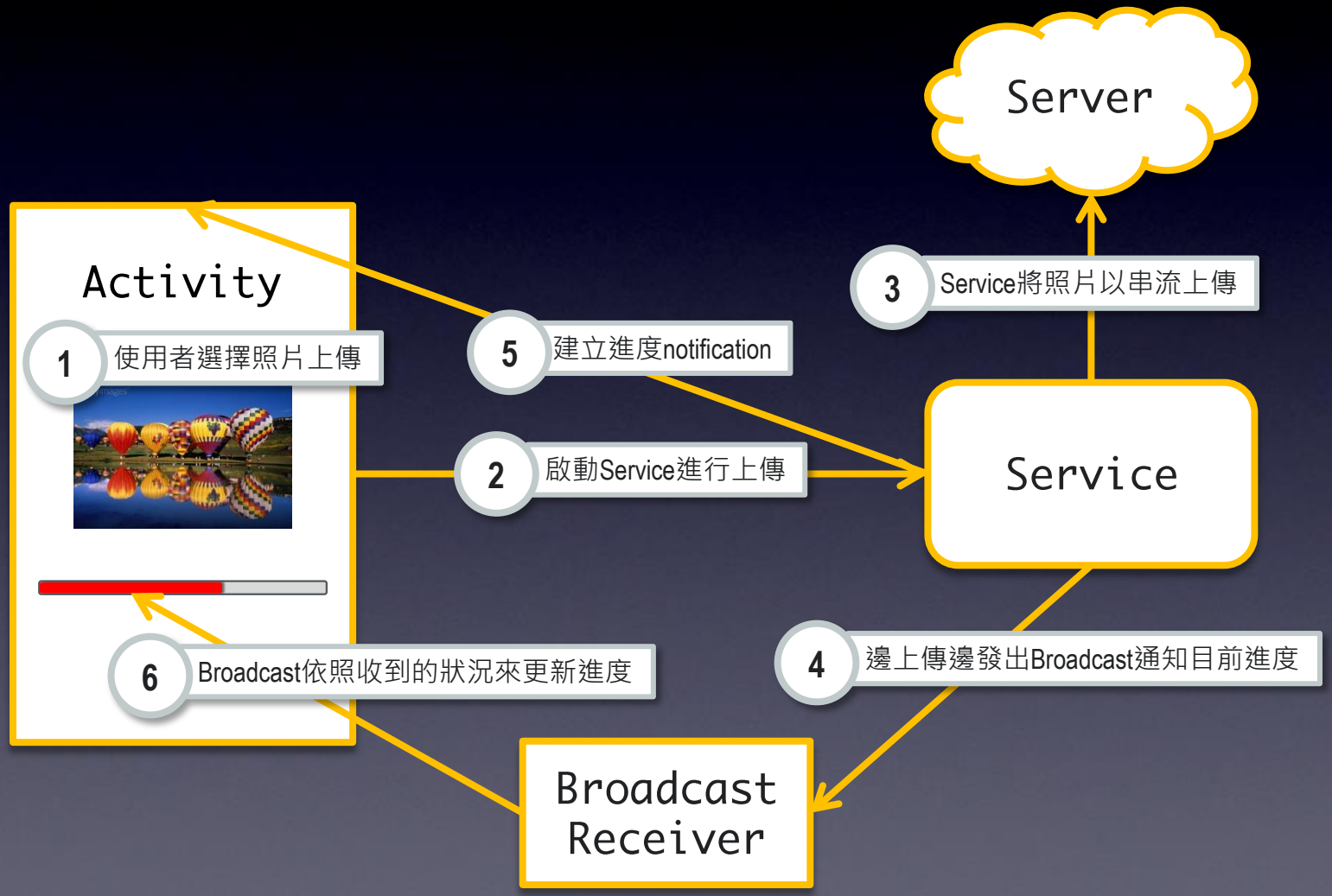
`stopService(Intent)`

- 一般來說Service不需要使用的時候需要呼叫 stopService()
- 但是使用IntentService不用考慮這一點，因為Android系統會自動管理

Service/StartServiceUpdateProgress

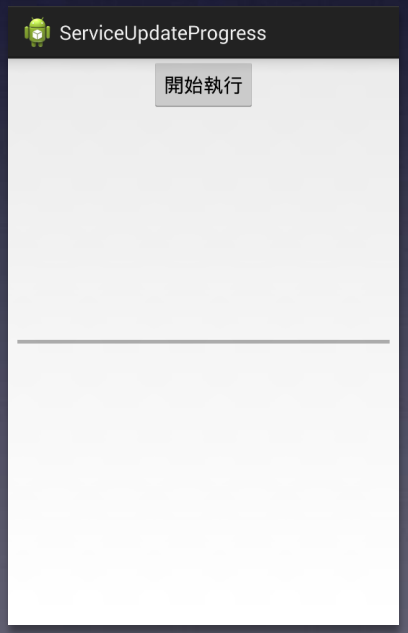
使用BROADCAST溝通

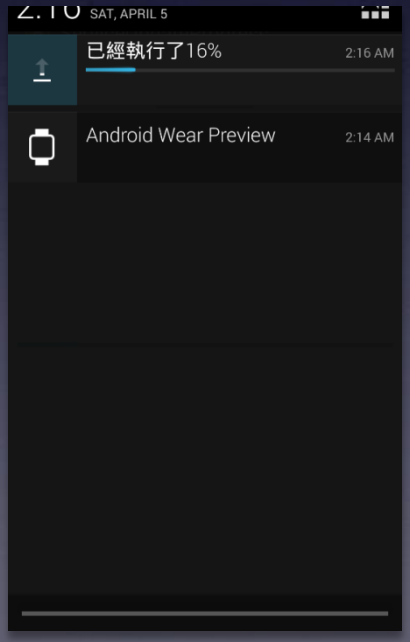
上傳下載常見作法

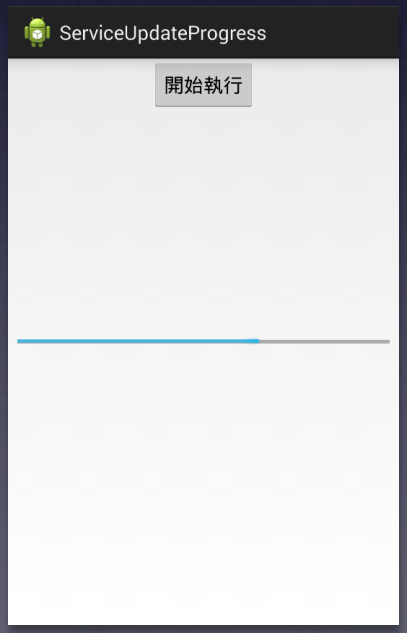

Server
Activity
1
使用者選擇照片上傳
5
建立進度notification
3
Service將照片以串流上傳
2
啟動Service進行上傳
Service
6
Broadcast依照收到的狀況來更新進度
4
邊上傳邊發出Broadcast通知目前進度
Broadcast Receiver

範例說明

- 以按下按鈕開啟上傳的Service
- Service啟動後會持續更新Notification
- Notification會顯示目前進度
- Activity的ProgressBar會依照Service發出的Broadcast更新







建立SERVICE

```
public class UpdateService extends IntentService {
	public UpdateService() {
		super("UpdateService");
	}
	…
```

- 首先讓class繼承IntentService
- 建立預設的建構子

建立SERVICE

```
protected void onHandleIntent(Intent intent) {
        int currentProgress = 0;
        while (currentProgress < 100) {
                currentProgress++;
                Intent updateIntent = new
                        Intent("com.example.serviceupdate.UPDATE");
                updateIntent.putExtra("progress", currentProgress);
                sendBroadcast(updateIntent);
                showNotification(currentProgress);
                ……
        }
}
```

- 在onHandleIntent處理外部啟動此Service的Intent

建立SERVICE

```
protected void onHandleIntent(Intent intent) {
        int currentProgress = 0;
        while (currentProgress < 100) {
                currentProgress++;
                Intent updateIntent = new
                        Intent("com.example.serviceupdate.UPDATE");
                updateIntent.putExtra("progress", currentProgress);
                sendBroadcast(updateIntent);
                showNotification(currentProgress);
                ……
        }
}
```

- 建立Intent，藉由sendBroadcast()發出Broadcast通知Receiver

建立持續更新的通知

```
private void showNotification(int progress) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_BROUGHT_TO_FRONT);
       PendingIntent contentIntent =
                    PendingIntent.getActivity(this, 0, intent,
                                PendingIntent.FLAG_UPDATE_CURRENT);
        ……
```

- 在showNotification的部分，建立PendingIntent

建立持續更新的通知

```
Notification notification;
if (builder == null)
        builder = new NotificationCompat.Builder(this);
builder.setContentIntent(contentIntent);
```

- builder是NotificationCompat.Builder的實體
- 保持builder是成員變數而非區域變數，要做到持續更新的notification，builder必須得是同一個

建立持續更新的通知

```
if (progress < 100) {
        builder.setContentTitle("已經執行了" + progress + "%")
                .setProgress(100, progress, false);
                .setSmallIcon(android.R.drawable.stat_sys_upload)
                .setOngoing(true);
        notification = builder.build();
} else {
        ……
}
```

- Android 4.0 (Ice Cream Sandwich)版之後，要讓Notification顯示ProgressBar，只要呼叫setProgress()即可
  - 參數1：progress最大值
  - 參數2：目前的progress
  - 參數3：ProgressBar是否會顯示當前進度

建立持續更新的通知

```
if (progress < 100) {
        builder.setContentTitle("已經執行了" + progress + "%")
                .setProgress(100, progress, false);
                .setSmallIcon(android.R.drawable.stat_sys_upload)
                .setOngoing(true);
        notification = builder.build();
} else {
        ……
}
```

- setOngoing()設定Notification為正在進行的類型

建立持續更新的通知

```
if (progress < 100) {
        ……
} else {
        builder
        .setSmallIcon(android.R.drawable.stat_sys_upload_done)
        .setOngoing(false)
        .setContentTitle("執行完成");
        notification = builder.build();
}
```

- 記得當處理完畢，要設定setOngoing()為false

舊版通知顯示進度條

- setProgress()目前只對android 4.0之後的有用
- 較舊的Android版本必須得自己建立RemoteViews

舊版通知顯示進度條

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id=“@+id/progress” …/>
    <ProgressBar
        android:id=“@+id/progress_bar”
        style=“?android:attr/progressBarStyleHorizontal” …/>
</LinearLayout>
```

- id progress負責以文字顯示進度
  id progress_bar負責顯示ProgressBar
- style=“?android:attr/progressBarStyleHorizontal”表示ProgressBar呈現進度條的模式

舊版通知顯示進度條

```
if (android.os.Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.
ICE_CREAM_SANDWICH) {
        RemoteViews views = new RemoteViews(getPackageName(),
                        R.layout.update_progress);
        views.setProgressBar(R.id.progress_bar,100,progress,false);
        views.setTextViewText(R.id.progress,"已經執行了"+progress+"%");
        notification = builder.build();
        notification.contentView = views;
}
```

- 在Service的showNotification中，可以用android.os.Build.VERSION.SDK_INT判斷系統版本
- 對應的版本編號可以用android.os.Build.VERSION_CODES來取得

舊版通知顯示進度條

```
if (android.os.Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.
ICE_CREAM_SANDWICH) {
        RemoteViews views = new RemoteViews(getPackageName(),
                        R.layout.update_progress);
        views.setProgressBar(R.id.progress_bar,100,progress,false);
        views.setTextViewText(R.id.progress,"已經執行了"+progress+"%");
        notification = builder.build();
        notification.contentView = views;
}
```

- 首先建立RemoteViews
- RemoteViews.setProgressBar()可以用來更新進度
  - 參數1: ProgressBar在RemoteViews中的id
  - 參數2: ProgressBar的最大值
  - 參數3: 目前的進度
  - 參數4: 進度條是否呈現目前進度

舊版通知顯示進度條

```
if (android.os.Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.
ICE_CREAM_SANDWICH) {
        RemoteViews views = new RemoteViews(getPackageName(),
                        R.layout.update_progress);
        views.setProgressBar(R.id.progress_bar,100,progress,false);
        views.setTextViewText(R.id.progress,"已經執行了"+progress+"%");
        notification = builder.build();
        notification.contentView = views;
}
```

- 最後藉由build()建立notification
- 但有ProgressBar類型的RemoteViews得使用notification.contentView = RemoteViews來指定，否則在舊版的手機上會看不到進度條

ACTIVITY部分

```
Intent intent = new Intent(MainActivity.this, UpdateService.class);
startService(intent);
```

- 在Activity部分使用Intent啟動Service

ACTIVITY部分

```
IntentFilter filter = new IntentFilter();
filter.addAction("com.example.serviceupdate.UPDATE");
registerReceiver(mReceiver, filter);
```

- 使用registerReceiver()在Activity啟動時註冊Broadcast Receiver負責接收Service的更新

ACTIVITY部分

```
protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(mReceiver);
}
```

- onDestroy()時記得取消Broadcast Receiver的註冊

BROADCAST RECEIVER部分

```
public BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context c, Intent intent) {
                int progress = intent.getIntExtra("progress", 0);
                mProgressBar.setProgress(progress);
        }
};
```

- 從Broadcast Receiver的onReceive()接收到Service發出的通知，從Intent中取得Service目前所更新道的進度

更多資料

更多資料

- 與App綁在一起的背景服務屬於進階議題
- 可以參考以下資料
  - http://developer.android.com/guide/components/services.html
  - http://developer.android.com/guide/components/aidl.html
  - http://developer.android.com/reference/android/os/Parcelable.html
- 參考專案
  - Service/BindServiceBasic
  - Service/BindServicePassObject
  - Service/BindServiceUpdateProgress