一、首先，讓我們確認下什麼是service？

service就是android系統中的服務，它有這麼幾個特點：它無法與用戶直接進行交互、它必須由用戶或者其他程序顯式的啟動、它的優先級比較高，它比處於前台的應用優先級低，但是比後台的其他應用優先級高，這就決定了當系統因為缺少內存而銷毀某些沒被利用的資源時，它被銷毀的概率很小哦。

二、那麼，什麼時候，我們需要使用service呢？

我們知道，service是運行在後台的應用，對於用戶來說失去了被關注的焦點。這就跟我們打開了音樂播放之後，便想去看看圖片，這時候我們還不想音樂停止，這裡就會用到service；又例如，我們打開了一個下載鏈接之後，我們肯定不想瞪著眼睛等他下載完再去做別的事情，對吧？這時候如果我們想手機一邊在後台下載，一邊可以讓我去看看新聞啥的，就要用到service。

三、service分類：

一般我們認為service分為兩類，本地service和遠程service。

本地service顧名思義，那就是和當前應用在同一個進程中的service，彼此之間擁有共同的內存區域，所以對於某些數據的共享特別的方便和簡單；

遠程service：主要牽扯到不同進程間的service訪問。因為android的系統安全的原因導致了我們在不同的進程間無法使用一般的方式共享數據。在這裡android為我們提供了一個AIDL工具。（android interface description language）android接口描述語言。在後邊我們將會對其進行詳細的介紹。

四、service生命週期：

和Activity相比，service的生命週期已經簡單的不能再簡單了，只有onCreate()->onStart()->onDestroy()三個方法。

Activity中和service有關的方法：

startService(Intent intent)：啟動一個service

stopService(Intent intent) :停止一個service

如果我們想使用service中的一些數據或者訪問其中的一些方法，那麼我們就要通過下面的方法：

public boolean bindService(Intent intent, ServiceConnection conn, int flags) ；

public void unbindService(ServiceConnection conn);

intent是跳轉到service的intent，如 Intent intent = new Intent();

intent.setClass(this,MyService.class);

conn則是一個代表與service連接狀態的類，當我們連接service成功或失敗時，會主動觸發其內部的onServiceConnected或onServiceDisconnected方法。如果我們想要訪問service中的數據，可以在onServiceConnected()方法中進行實現，

使用service的步驟：

第一步：我們要繼承service類，實現自己的service。

如果想要訪問service中的某些值，我們通常會提供一個繼承了Binder的內部類，通過onBund()方法返回給service請求。這裡實際上巧妙的利用了內部類能夠訪問外部類屬性的特點。

第二步：在androidManifest.xml中進行註冊，如：

<!-- service配置開始 -->

<service android:name="MyService"></service>

<!-- service配置結束-->

第三步：在activity中進行啟動、綁定、解綁或者停止service。

（很多書上說，service與用戶是不能交互的，其實這話很不正確，我們完全可以通過activity與service進行交互！我認為，確切的說法應該是service與用戶不能進行直接的交互）。

---

bindService介紹

一、bindService簡介

bindService是綁定Service服務，執行service服務中的邏輯流程。

service通過Context.startService()方法開始，通過Context.stopService()方法停止；也可以通過Service.stopSelf()方法或者Service.stopSelfResult()方法來停止自己。只要調用一次stopService()方法便可以停止服務，無論之前它被調用了多少次的啟動服務方法。

客戶端建立一個與Service的連接，並使用此連接與Service進行通話，通過Context.bindService()方法來綁定服務，Context.unbindService()方法來關閉服務。多個客戶端可以綁定同一個服務，如果Service還未被啟動，bindService()方法可以啟動服務。

上面startService()和bindService()兩種模式是完全獨立的。你可以綁定一個已經通過startService()方法啟動的服務。例如：一個後台播放音樂服務可以通過startService(intend)對象來播放音樂。可能用戶在播放過程中要執行一些操作比如獲取歌曲的一些信息，此時activity可以通過調用bindServices()方法與Service建立連接。這種情況下，stopServices()方法實際上不會停止服務，直到最後一次綁定關閉。

- 如果沒有程序停止它或者它自己停止，service將一直運行。在這種模式下，service開始於調用Context.startService() ，停止於Context.stopService(). service可以通過調用Android Service 生命週期() 或 Service.stopSelfResult()停止自己。不管調用多少次startService() ，只需要調用一次 stopService() 就可以停止service。
- 可以通過接口被外部程序調用。外部程序建立到service的連接，通過連接來操作service。建立連接調開始於Context.bindService(), 結束於Context.unbindService(). 多個客戶端可以綁定到同一個service，如果service沒有啟動， bindService() 可以選擇啟動它。
- 這2種模式不是完全分離的。你可以可以綁定到一個通過startService()啟動的服務。如一個intent想要播放音樂，通過startService() 方法啟動後台播放音樂的service。然後，也許用戶想要操作播放器或者獲取當前正在播放的樂曲的信息，一個activity就會通過bindService()建立一個到此service的連接. 這種情況下 stopService() 在全部的連接關閉後才會真正停止service。

二、bindService啟動流程

context.bindService() ——> onCreate() ——> onBind() ——> Service running ——> onUnbind() ——> onDestroy() ——> Service stop

onBind()將返回給客戶端一個IBind接口實例，IBind允許客戶端回調服務的方法，比如得到Service的實例、運行狀態或其他操作。這個時候把調用者（Context，例如Activity）會和Service綁定在一起，Context退出了，Srevice就會調用onUnbind->onDestroy相應退出。

所以調用bindService的生命週期為：onCreate --> onBind(只一次，不可多次綁定) --> onUnbind --> onDestory。

在Service每一次的開啟關閉過程中，只有onStart可被多次調用(通過多次startService調用)，其他onCreate，onBind，onUnbind，onDestory在一個生命週期中只能被調用一次。

三、bindService生命週期

像一個activity那樣，一個service有些可以用來改變狀態的生命週期方法，但是比activity的方法少，service生命週期方法只有三個public

void onCreate()

void onStart(Intent intent)

void onDestroy()

通過實現這三個生命週期方法，你可以監聽service的兩個嵌套循環的生命週期：

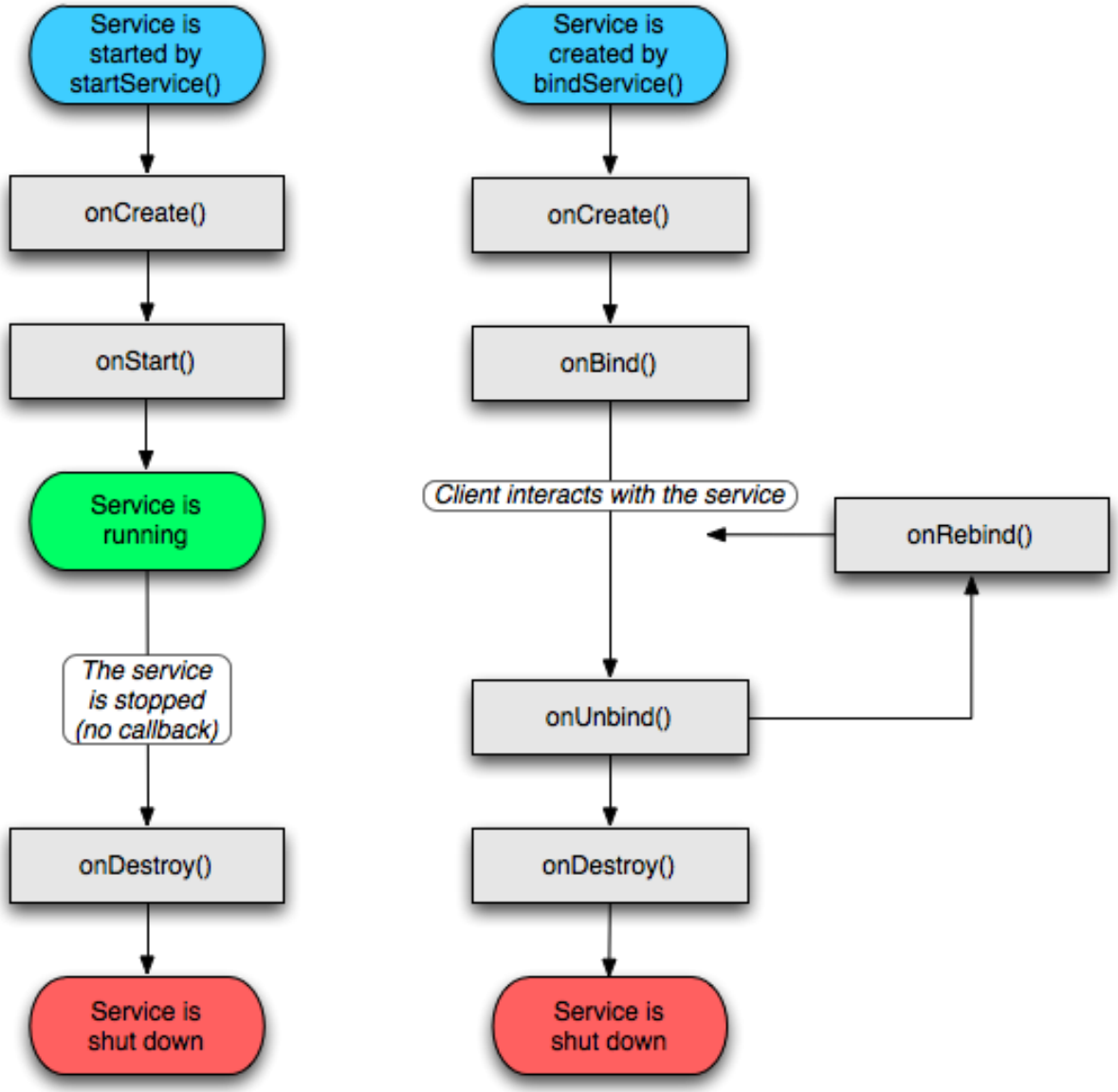
1、整個生命週期

service的整個生命週期是在onCreate()和onDestroy()方法之間。和activity一樣，在onCreate()方法裡初始化，在onDestroy()方法裡釋放資源。例如，一個背景音樂播放服務可以在onCreate()方法裡播放，在onDestroy()方法裡停止。

2、活動的生命週期

service的活動生命週期是在onStart()之後，這個方法會處理通過startServices()方法傳遞來的Intent對象。音樂service可以通過開打intent對象來找到要播放的音樂，然後開始後台播放。註： service停止時沒有相應的回調方法，即沒有onStop()方法，只有onDestroy()銷毀方法。

onCreate()方法和onDestroy()方法是針對所有的services，無論它們是否啟動，通過Context.startService()和Context.bindService()方法都可以訪問執行。然而，只有通過startService()方法啟動service服務時才會調用onStart()方法。



如果一個service允許別人綁定，那麼需要實現以下額外的方法：

IBinder onBind(Intent intent)

boolean onUnbind(Intent intent)

void onRebind(Intent intent)

onBind()回調方法會繼續傳遞通過bindService()傳遞來的intent對象

onUnbind()會處理傳遞給unbindService()的intent對象。如果service允許綁定，onBind()會返回客戶端與服務互相聯繫的通信句柄（實例）。

如果建立了一個新的客戶端與服務的連接，onUnbind()方法可以請求調用onRebind()方法。

記住： 任何服務無論它怎樣建立，默認客戶端都可以連接，所以任何service都能夠接收onBind()和onUnbind()方法

四、bindService和startservice示例

（1）mainactivity

```
public class MainActivity extends Activity {
    Button startServiceButton;// 啟動服務按鈕
    Button shutDownServiceButton;// 關閉服務按鈕
    Button startBindServiceButton;// 啟動綁定服務按鈕
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getWidget();
        regiestListener();
    }

    /** 獲得組件 */
    public void getWidget() {
        startServiceButton = (Button) findViewById(R.id.startServerButton);
        startBindServiceButton = (Button) findViewById(R.id.startBindServerButton);
        shutDownServiceButton = (Button) findViewById(R.id.sutdownServerButton);
    }

    /** 為按鈕添加監聽 */
    public void regiestListener() {
        startServiceButton.setOnClickListener(startService);
        shutDownServiceButton.setOnClickListener(shutdownService);
        startBindServiceButton.setOnClickListener(startBinderService);
```

```
    }


    /** 啟動服務的事件監聽 */
    public Button.OnClickListener startService = new Button.OnClickListener() {
        public void onClick(View view) {
            /** 單擊按鈕時啟動服務 */
            Intent intent = new Intent(MainActivity.this,
                    CountService.class);
            startService(intent);

            Log.v("MainStadyServics", "start Service");
        }
    };
    /** 關閉服務 */
    public Button.OnClickListener shutdownService = new Button.OnClickListener() {
        public void onClick(View view) {
            /** 單擊按鈕時啟動服務 */
            Intent intent = new Intent(MainActivity.this,
                    CountService.class);
            /** 退出Activity是，停止服務 */
            stopService(intent);
            Log.v("MainStadyServics", "shutDown serveice");
        }
    };
    /** 打開綁定服務的Activity */
    public Button.OnClickListener startBinderService = new Button.OnClickListener() {
        public void onClick(View view) {
            /** 單擊按鈕時啟動服務 */
            Intent intent = new Intent(MainActivity.this, UseBrider.class);
            startActivity(intent);
            Log.v("MainStadyServics", "start Binder Service");
        }
    };
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
```

```
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
```

（2）service

```
package com.example.testservice;

/**引入包*/
import android.app.Service;// 服務的類
import android.os.IBinder;
import android.os.Binder;
import android.content.Intent;
import android.util.Log;

/** 計數的服務 */
public class CountService extends Service {
    /** 創建參數 */
    boolean threadDisable;
    int count;

    public IBinder onBind(Intent intent) {
        return null;
    }

    public void onCreate() {
        super.onCreate();
        /** 創建一個線程，每秒計數器加一，並在控制台進行Log輸出 */
        new Thread(new Runnable() {
            public void run() {
                while (!threadDisable) {
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
```

```
                }
                count++;
                Log.v("CountService", "Count is" + count);
            }
        }
    }).start();
  }

  public void onDestroy() {
    super.onDestroy();
    /** 服務停止時，終止計數進程 */
    this.threadDisable = true;
  }

  public int getConunt() {
    return count;
  }

//此方法是為了可以在Acitity中獲得服務的實例
class ServiceBinder extends Binder {
    public CountService getService() {
        return CountService.this;
    }
  }
}
```

（3）bindservice（一定要記著這個是要獲得，鏈接的對象）

```
package com.example.testservice;

/**引入包*/
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;

/** 通過bindService和unBindSerivce的方式啟動和結束服務 */
public class UseBrider extends Activity {
   /** 參數設置 */
   CountService countService;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(new UseBriderFace(this));

      Intent intent = new Intent(UseBrider.this, CountService.class);
      /** 進入Activity開始服務 */
      bindService(intent, conn, Context.BIND_AUTO_CREATE);

   }


   private ServiceConnection conn = new ServiceConnection() {
      /** 獲取服務對象時的操作 */
      public void onServiceConnected(ComponentName name, IBinder service) {
         // TODO Auto-generated method stub
         countService = ((CountService.ServiceBinder) service).getService();

      }

      /** 無法獲取到服務對象時的操作 */
      public void onServiceDisconnected(ComponentName name) {
         // TODO Auto-generated method stub
         countService = null;
      }
```

```
    };

    protected void onDestroy() {
        super.onDestroy();
        this.unbindService(conn);
        Log.v("MainStadyServics", "out");
    }
}
```

注意：這個地方有朋友可能會出現onServiceConnected不調用的情況。

這個問題當調用bindService方法後就會回調Activity的onServiceConnected，在這個方法中會向Activity中傳遞一個IBinder的實例，Acitity需要保存這個實例

在Service中需要創建一個實現IBinder的內部類(這個內部類不一定在Service中實現，但必須在Service中創建它)。

在OnBind（）方法中需返回一個IBinder實例，不然onServiceConnected方法不會調用。

不過，我在這裡傳遞null也能夠調用，大家根據情況進行判定吧，如果是返回一個ibinder實例的話，示例代碼如下：

```
public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        System.out.println("onBind.....");
         IBinder result = null;
            if ( null == result ) result = new MyBinder() ;
        Toast.makeText(this, "onBind",Toast.LENGTH_LONG);
        return result;
    }
```

至於startservice和bindservice的使用場景，有網友這麼說：

1.通過startservice開啟的服務.一旦服務開啟, 這個服務和開啟他的調用者之間就沒有任何的關係了.

調用者不可以訪問 service裡面的方法. 調用者如果被系統回收了或者調用了ondestroy方法, service還會繼續存在

2.通過bindService開啟的服務,服務開啟之後,調用者和服務之間 還存在著聯繫 ,

一旦調用者掛掉了.service也會跟著掛掉 .

示例下載地址：http://pan.baidu.com/share/link?shareid=1614272126&uk=1428765741

還有一個多樣化的demo學習地址：

http://pan.baidu.com/share/link?shareid=1616100229&uk=1428765741