Service 和 Broadcast Receiver， intent

使用说明

【源代码根目录】： 工程目录 ServiceIntentIReceiver

【记号】:

(@编程练习)：表明该实验是需要编将工程文件和实验报告一起提交。

(@团队编程练习):表明该实验是以小组为单位完成的，每个小组完成一份程序和报告即可，报告和工程和其他的实验报告和工程最后要分开交。

本次实验的个人编程部分，请建立在名为 KH7 的 project 中。上传方式：

http://disk.lehu.shu.edu.cn/index.aspx

上传码：

android-sybk (本科)

android-syyjs（研究生）

【实验报告的要求】

1、 文件名规范： 学号+姓名+实验名称.doc

2、 内容格式见 实验报告格式.doc

【工程提交的要求】

目录

【课堂实验 KT7.1】：Service 生命周期

【工程模块】：ServiceBaseSample
【工程描述】：该工程展示了启动和停止 service 的方法；绑定和解绑 service 的方法；以及 service 各个生命周期的变化。
【实验步骤】：运行程序，点击各个按钮，观察输出。
改变点击按钮的顺序，观察现象，理解 service 的生命周期以及创建和销毁的过程。
【知识点注释】：

Service 是四大构件之一（Activity、Service、BroadcastReceiver、ContentProvider），用于执行无需 UI 的后台任务。比如在后台播放音乐或者下载文件。和在 Activity 中开启一个线程来执行相比，任务放在 Service 中执行最主要的区别在于，系统会更优先保持 Service 的进程，而 Activity 中的线程可能会因为 UI 切换而被系统终止。使用 Service 还有如下几点要特别知晓：

1、 同 Activity 一样，Service 需要在配置文件中声明。
2、 Service 是运行在主线程中的，其宿主进程并不创建一个新的线程来运行 Service，
因此 Service 不应该执行耗时操作，否则也会引发 ANR（Application Not Responding）错误 。
3、 Service 可以使用两种方法启动：
   a) startService()
   b) bindService()
4、 使用 startService()启动的 Service，需要用 stopService 或者 stopSelf 停止，而用 bindService 启动的 Service 则可以用 unbindServcie 解除绑定。
5、 仅仅支持 startService 启动的 Service 应该实现 onStartCommand()回调函数，而仅仅支持 bindService 启动的 Servcie 应该实现 onBind()回调函数。

Service 和 Activity 一样，启动时需要传递一个 intent 对象，可以显式也可隐形启动（注意注释的部分是显式方式启动）：

```
public void bt1Click(View view){
    Intent it=new Intent(MyService.ACTION);
    it.setPackage("com.example.me.servicebasesample");
   // Intent it=new Intent(this,MyService.class);
    startService(it);
}

public void bt2Click(View view){
    Intent it=new Intent(MyService.ACTION);
    it.setPackage("com.example.me.servicebasesample");
   // Intent it=new Intent(this,MyService.class);
    stopService(it);
}

public void bt3Click(View view){
    Intent it=new Intent(MyService.ACTION);
    it.setPackage("com.example.me.servicebasesample");
    // Intent it=new Intent(this,MyService.class);
    bindService(it, conn, Context.BIND_AUTO_CREATE);
    mBound = true;
}

public void bt4Click(View view){
    if(  mBound) {
        unbindService(conn);
        mBound = false;
    }
}
```

注意隐式启动时，应该设置包名: Intent it=new Intent(MyService.ACTION);
it.setPackage("com.example.me.servicebasesample");

【课堂实验 KT7.2】: bind Service

【工程路径】: ServiceRemoteSample|ServiceRemoteClientSample
【工程描述】: 该例展示了 activity 如何通过绑定 service 来远程调用 service 中的方法来和 service 交互。
【实验步骤】: 先运行 ServiceRemoteSample，将有关 service 安装到目标机器。每次点击按钮可看到获得一个随机数。运行 ServiceRemoteClientSample,点击按钮查看结果。
【知识点注释】:
本例展示了两种绑定 service 地方法，其中实现了一个跨进程的远程 bind 方法，因此需要用

两个 module 来实现。一个负责实现 service，一个则是 client。第一个实现了两个 service，一个是 LocalService，一个是 MessengerService。请注意 AndroidManifest 中的声明。

```
public class LocalService extends Service {
    // Binder given to clients
    private final IBinder mBinder = new LocalBinder();
    // Random number generator
    private final Random mGenerator = new Random();

    /**
     * Class used for the client Binder.  Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // Return this instance of LocalService so clients can call public methods
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    /** method for clients */
    public int getRandomNumber() {
        return mGenerator.nextInt(100);
    }
```

```
public class MessengerService extends Service {
    /** Command to the service to display a message */
    static final int MSG_SAY_HELLO = 1;
    public static final String ACTION="com.example.me.serviceremotesample_ACTION";

    /**
     * Handler of incoming messages from clients.
     */
    class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MSG_SAY_HELLO:
                    Toast.makeText(getApplicationContext(), "hello!, from MessengerService",
Toast.LENGTH_SHORT).show();
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }

    /**
     * Target we publish for clients to send messages to IncomingHandler.
     */
    final Messenger mMessenger = new Messenger(new IncomingHandler());

    /**
     * When binding to the service, we return an interface to our messenger
```

【课堂实验 KT7.3】：BroadcastReceiver

【工程名】ReceiverSample
【工程描述】：该例展示了自定义 receiver 的基本用法
【知识点注释】：
Receiver 用于监听构件之间以及构件与系统之间发送的消息。唯一要实现的回调函数是 onReceive()。注意不要再该函数中执行耗时操作，如果需要如此，应在其中开启一个 service 来进行。

```
public class MyReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context cxt, Intent intent) {

        String msg = intent.getStringExtra("msg");

        Toast.makeText(cxt,"土豆土豆，我是地瓜", Toast.LENGTH_LONG).show();

        NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(cxt)
                        .setSmallIcon(R.mipmap.ic_launcher)
                        .setContentTitle("My notification")
                        .setContentText(msg);

        NotificationManager mNotificationManager =
                (NotificationManager) cxt.getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
        mNotificationManager.notify(1, mBuilder.build());
    }
}
}
```

.....发送广播的代码

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 实例化 Intent 对象
        Intent intent = new Intent();
        // 设置 Intent action 属性
        intent.setAction(MY_ACTION);
        // 为 Intent 添加附加信息
        intent.putExtra("msg",tv.getText());
        // 发出广播
        sendBroadcast(intent);
    }
```

【课堂实验 KT7.4】：通知 Notification 的用法

【工程路径】： NotificationSample
【工程描述】:
本工程有三个 activity。
Mainactivity:有三个按钮，分别展示创建 big view 风格的 notification。以及两种风格进度条的 notification。
Resultactivity：分别展示了创建包含 backstack pendingintent 的通知和 special pendingintent 的通知
Resultsingleactivity:展示了如何从 notification 中获得信息，以及如何清除 notification。
【知识点注释】:
Notification

基本用法

一、创建 notification

1、 使用 NotificationCompat.Builder 类可以定制 notification 的 UI 和操作
2、 使用 NotificationCompat.Builder.build()方法可以返回这个 Notification 对象，若要创建 Big view style，可以调用 Builder.setStyle()。
3、 调用 NotificationManager.notify()方法将 Notification 最终提交给系统.

Notification 对象必须设置如下内容，其余的部分则可选:

- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- Detail text, set by setContentText()

二、Notification 的 Action

可以为 Notification 对象设置一些操作（Aciton），比如用户点击 Notification 时，通常会激活一个操作将用户带到该 Notification 相关的 Activity 中。所打开的 activity 的 back stack 的行为有两种设置方法。

Setting up a regular activity PendingIntent

```
<activity
      android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER"
/>
      </intent-filter>
</activity>
<activity
      android:name=".ResultActivity"
      android:parentActivityName=".MainActivity">
      <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".MainActivity"/>
</activity>
```

Setting up a special activity PendingIntent

```
<activity
      android:name=".ResultActivity"
...
      android:launchMode="singleTask"
      android:taskAffinity=""
      android:excludeFromRecents="true">
</activity>
```

三、创建包含进度条的 notification

版本 4.1 以上可以使用 setProgress() 函数直接获得包含一个进度条的 notification。而早期版本则要通过定制 notification 的 layout 来实现。

【课后实验 KH7.1】

【功能要求】:（@编程练习）通常通过 startService 启动 Service 意味着启动者无需也无法同 service 通信，但我们可以通过 receiver 让 activity 和 service 进行交互，请设计这个程序，activity 通过 startService 启动一个 service，然后发送广播向 service 请求数据（比如一个随机数），service 收到请求后也发送广播给 activity 回传数据。功能和界面可自行定义。