



河北师范大学软件学院
Software College of Hebei Normal University



Android 基础开发

第十章 Android的Service



Java与移动智能设备开发



教学目标

- 掌握Android中Service的使用



目录

1

Service简介

2

Service的使用

3

跨进程调用Service

4

常见系统服务调用



Service简介



- Android中经常会有这样一种程序？





Service简介

- 服务是指执行指定系统功能的程序、例程或进程，以便支持其它程序，尤其是底层(接近硬件)程序。我们看不到他们在运行，但是它却在默默为我提供者服务。



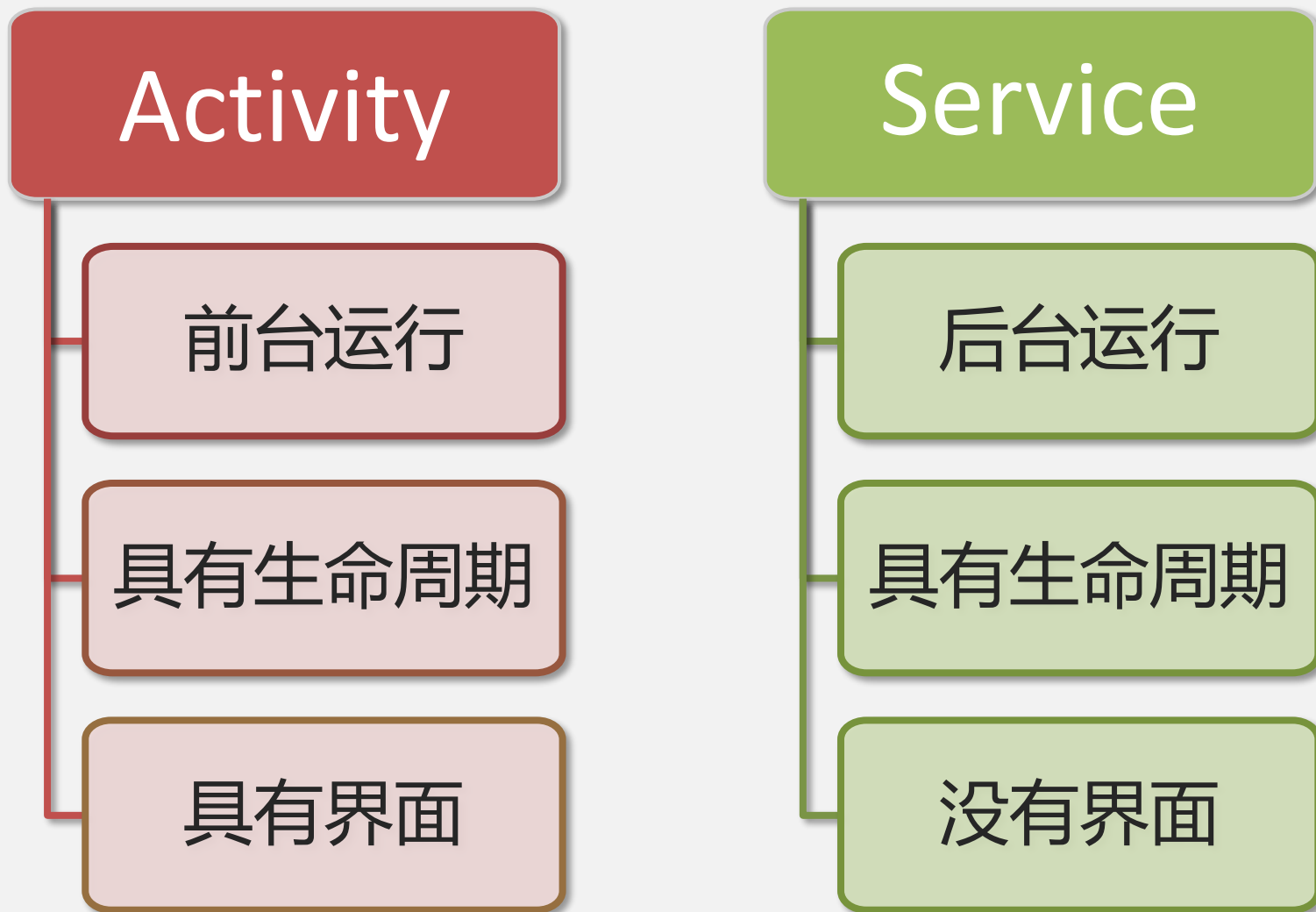


目录

- 1 Service简介
- 2 Service的使用**
- 3 跨进程调用Service
- 4 常见系统服务调用



Service VS Activity



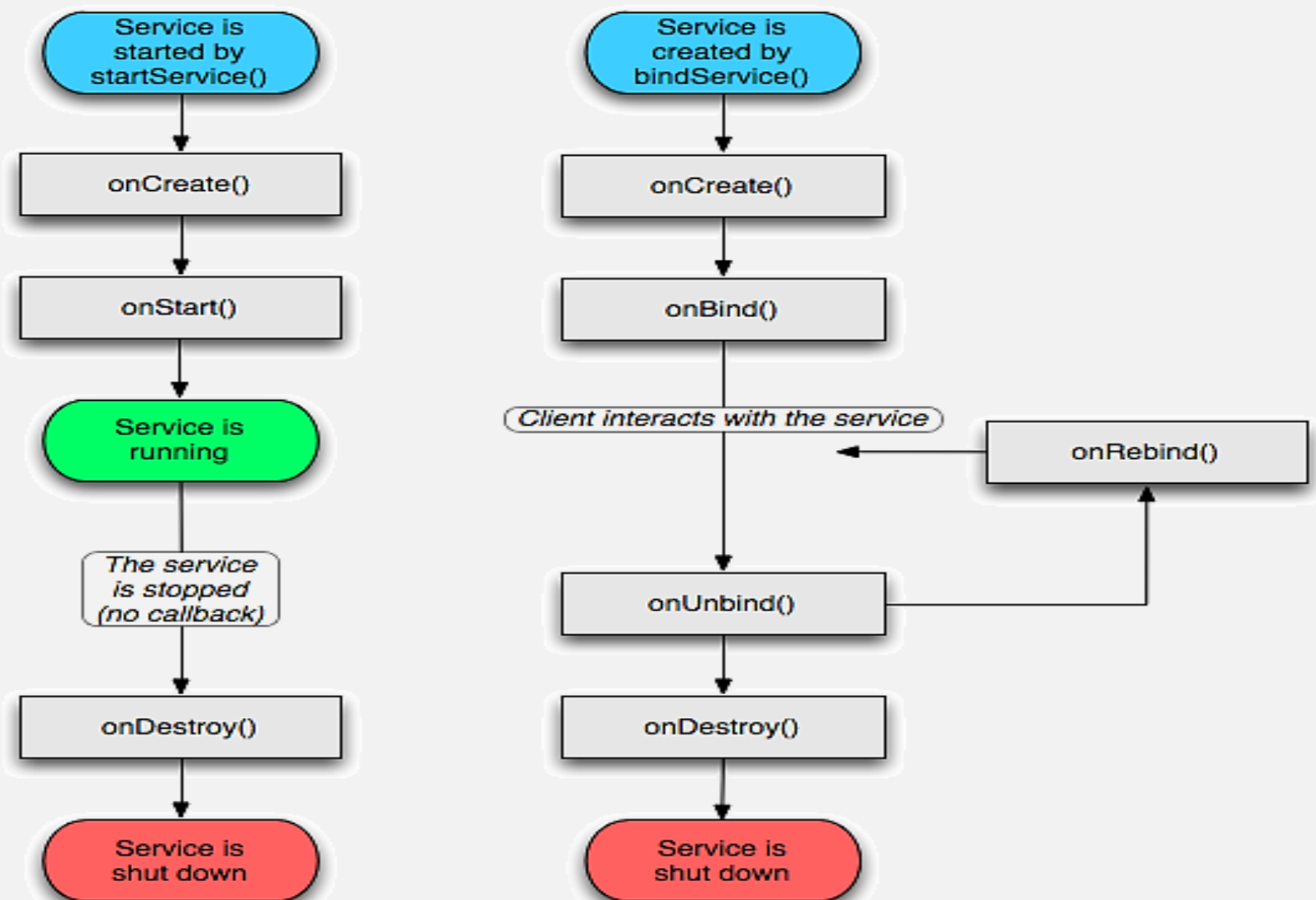


Service的使用

- 在Android中创建Service的步骤：
 - 定义一个类，继承自Service，实现相应的接口。
 - 在AndroidManifest.xml中进行Service的注册。
- Service和Activity都是继承自Context的，都可以直接调getResources，getContentResolver等方法。



Service的生命周期





Service的生命周期

- Service生命周期中的常见回调函数：
 - abstract IBinder **onBinder**(intent)：必须实现，应用程序通过IBinder对象与service通讯。
 - void **onCreadte**()：创建时回调。
 - void **onDestroy**()：关闭前回调。
 - void **onStartCommand**(intent,flags,startID)：startService被调用时回调该方法。
 - boolean **onUnbind**(intent)：Service绑定的所有客户端都断开时回调该方法。



Service的注册

- 在Android中注册Service：
 - 打开AndroidManifest.xml。
 - 在Application节点中加入Service节点。

```
<!-- 配置一个Service组件 --!>
<service android:name = ".FirstService"
    <intent-filter>
        <!-- 配置service的filter属性 --!>
        <action android:name="过滤器串儿" />
    </intent-filter>
</service>
```



Service的调用

- 在Android提供两种Service的调用方式：
 - 通过Context的**startService()**方法：访问者与Service之间没有关联，即使访问者退出了，Service依然运行。
 - 通过Context的**bindService()**方法：访问者与Service绑定在一起，访问者退出，Service也就停止。



Service的启动和停止

- 通过Activity访问Service，Demo(Android5.0 以后，必须使用显示Intent启动Service组件)。
- Activity和Service之间没有关系，所以无法进行方法调用或者数据交换。



绑定本地Service

- 通过bindService方法绑定Service：
 - **service**：通过intent启动的service。
 - **conn**：ServiceConnection对象，通过一系列的回调函数来监听访问者和Service的连接情况。
 - **flags**：绑定时是否自动创建Service。
- 访问者通过ServiceConnection对象的，onServiceConnected方法的IBinder对象来实现与Service的交互。
- 通常采用自定义类继承IBinder类来实现。



绑定本地Service

- 通过绑定方式连接Service

```
@Override
public IBinder onBind(Intent intent) {
    // 返回IBinder对象
    return binder;
}
private ServiceConnection conn = new ServiceConnection(){
    // 当该Activity与Service连接成功时回调该方法
    @Override
    public void onServiceConnected(ComponentName name,
                                   IBinder service) {
        // 获取Service的onBind方法所返回的MyBinder对象
        binder = (BindService.MyBinder) service;
    }
};
```



绑定本地Service

- 通过绑定方式连接Service

```
getServiceStatus.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View source) {  
        // 获取、并显示Service的count值  
        Toast.makeText(MainActivity.this,  
            "Service的count值为：" + binder.getCount(),  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

对于Service的onBind()方法所返回的IBinder对象来说，它可被当成该Service组件所返回的代理对象，Service允许客户端通过该IBinder对象来访问Service内部的数据，这样既可实现客户端与Service之间的通信。



绑定本地Service

- 当程序调用unbind方法接受对于某个Service的绑定时，系统会先回调Service的onUnbind方法，然后回调onDestroy方法。
- 注意：
 - 多次调用startService不会再调用onCreat方法。
 - 多次调用bindService对同一个service绑定一次。



IntentService

- Service存在问题：
 - Service和所在应用位于同一个进程中。
 - Service不是新线程，不能直接处理耗时的任务。
- IntentService是Service的子类，使用队列管理请求的Intent，使用worker线程同一时刻只处理一个Intent请求，不会阻塞主线程，可自己处理耗时任务。



IntentService

- IntentService的特征：
 - 创建单独的worker线程处理Intent请求。
 - 创建单独的worker线程处理onHandleIntent()方法，无需考虑多线程问题。
 - 所以请求完成后，IntentService自动停止。
 - 需重写onHandleIntent()方法，无需重写onBind()、onStartCommand()方法。



IntentService

- IntentService的使用步骤
 1. 自定义Service子类，继承IntentService类。
 2. 重写onHandleIntent()方法。
 3. 在Activity中定义Intent并启动IntentService（startService方法）。



IntentService

```
public class MyIntentService extends IntentService {  
    // IntentService会使用单独的线程来执行该方法的代码  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        // 该方法内可以执行任何耗时任务，比如下载文件等，此处只是让线程暂停20秒  
        long endTime = System.currentTimeMillis() + 20 * 1000;  
        while (System.currentTimeMillis() < endTime) {  
            synchronized (this) {  
                try {  
                    wait(endTime - System.currentTimeMillis());  
                } catch (Exception e) {  
                }  
            }  
        }  
        // 耗时任务执行完成  
    }  
}
```



IntentService

```
public void startIntentService(View source) {  
    // 创建需要启动的IntentService的Intent  
    Intent intent = new Intent(this, MyIntentService.class);  
    // 启动IntentService  
    startService(intent);  
}
```



目录

- 1 Service简介
- 2 Service的使用
- 3 跨进程调用Service**
- 4 常见系统服务调用



跨进程调用Service

- Android系统中，每个应用程序都运行在自己的进程中，进程间一般**无法直接进行数据交换**。
- 远程Service调用一般都是通过定义一个远程调用接口，然后为该接口实现一个类即可。客户端访问此Service时将一个回调对象Ibinder通过onBind方法返回给客户端。



跨进程调用Service

- 与绑定本地Service不同的是，本地Service的onBind方法会直接把Ibinder实现类给客户端的ServiceConnection，但远程Service的onBind方法只是将Ibinder对象的代理传给客户端的ServiceConnection。
- 客户端通过Ibinder的代理去访问Service的属性。
- Android中使用AIDL实现远程Service接口。



跨进程调用Service

- AIDL是Android Interface Definition Language的缩写，它是一种Android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口。
- AIDL语言的语法：
 - AIDL定义接口的源代码必须以.aidl结尾。
 - AIDL接口中的数据类型，有基本类型，String，List，Map，CharSequence之外都需要导入包，即使他们在同一个包内。



跨进程调用Service

- 跨进程调用Service的步骤：
 1. 创建AIDL文件。
 2. 将接口暴露给客户端。
 3. 客户端访问aidl service。



Step1、创建AIDL文件

- AIDL文件的创建：
 - Service端，客户端都需要通过Android SDK中的aidl.exe工具为该接口提供实现。
 - 使用eclipse+ADT，由ADT自动为AIDL接口产生实现。
- 例如：
 - 定义好Icat.aidl后，ADT会自动生成ICat.java文件，其中包含一个Stub内部类，此类实现了Ibinder，Icat两个接口，Stub类作为回调类。



Step2、将接口暴露给客户端

- 定义一个Service实现类，其onBind方法返回的Ibinder对象是ADT生成的ICat.Stub的子类的实例。
- 在AndroidManifest.xml文件中注册Service。

```
<!-- 定义一个Service组件 -->
<service android:name=".AidlService" >
    <intent-filter>
        <action android:name="包名" />
    </intent-filter>
</service>
```



Step3、客户端访问AIDLService



- 访问AIDLService的步骤：
 1. 讲Service端的AIDL接口文件复制到客户端应用中。
 2. 创建ServiceConnection对象。
 3. 调用Context.bindService()方法绑定远程Service。
 4. 在ServiceConnection对象的onServiceConnection方法中进行service对象返回值的处理。



目录

- 1 Service简介
- 2 Service的使用
- 3 跨进程调用Service
- 4 常见系统服务调用**



常见系统服务调用

- Android系统中很多系统已有的服务都已经在运行过程中，可以通过获得系统的服务管理器的方式进行系统服务的调用：
 - 电话管理器 (TelephonyManager)
 - 短信管理器 (SmsManager)
 - 音频管理器 (AudioManager)
 - 振动器 (Vibrator)
 - 手机闹钟服务 (AlarmManager)
 -



电话管理器 (TelephonyManager)



- TelephonyManager是一个管理手机通话状态、电话网络信息的服务类。
- 通过一系列的get方法得到电话通讯的一些信息。
- 获取手机位置、手机状态、读取通话状态需配置权限。

```
<!-- 添加访问手机位置的权限 -->
```

```
<uses-permission
```

```
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
<!-- 添加访问手机状态的权限 -->
```

```
<uses-permission
```

```
android:name="android.permission.READ_PHONE_STATE"/>
```



短信管理器 (SmsManager)



- SmsManager是Android中的短信服务，使用sendXxxMessage方法可以进行发送短信。
- 短信通常采用普通文本内容，可调用sendTextMessage()方法进行发送。



音频管理器 (AudioManager)



- AudioManager是Android用来管理音量的服务。通过getSystemService来获得音频管理器
 - getSystemService(Service.AUDIO_SERVICE)
- 其中常用方法：
 - adjustStreamVolume(int streamType, int direction, int flags) : 调整手机声音类型。
 - setMicrophoneMute(boolean on) : 是否麦克风静音。
 - SetRingerMode(int ringerMode) : 设置手机电话铃声的模式。
 - setSpeakerphoneOn(boolean on) : 是否打开扩音器。
 - setStreamVolume(int streamType, boolean state) : 设置手机指定类型的音量值。



振动器 (Vibrator)

- Vibrator是Android提供的对于系统的振动器的管理工具，同样他可以通过getSystemService的方法获得。
- 通过如下方法操纵震动器：
 - vibrate(long milliseconds)：控制手机震动m毫秒。
 - Vibrate(pattern , repeat)：间隔震动。
 - Cancel()：关闭手机震动。
- 需要控制手机振动的权限：

```
<users-permission  
android:name="android.permission.VIBRATE" >
```



手机闹钟服务 (AlarmManager)

- Alarm通常的用途是开发闹钟服务，他的本质是一个全局的定时器。可以实现定时完成某些任务。
- 通过getSystemService来获得AlarmManager对象。
通过如下方法执行任务：
 - Set(type,trigger,operation)：定时启动operation。
 - setRepeating()：设置周期任务。
 - cancel()：取消闹钟服务。



内容回顾

- 1 Service简介
- 2 Service的使用
- 3 跨进程调用Service
- 4 常见系统服务调用



Thank you!

