# Android移动开发课程

# 实验指导书-Service 部分

【实验目的】

初步了解Android 的 Service 组件

初步了解Android 的 多线程

【实验设计】

本次实验包括五个验证实验和一个自选实验。

其中验证实验为个人实验，已经提供源代码、操作步骤、实验指导视频

要求：

按照实验步骤完成，

以个人为单位提交，提交实验报告一份，实验报告需要回答指导书中问题。

自选实验为小组实验，建议2-4人组队，提供了参考选题和参考资料。

要求：

以小组为单位提交，提交实验报告一份、源代码一份、可以执行的APK文件一个。

【实验内容】

Service是一个Android中实现程序后台运行的解决方案。

Service是一种可在后台执行长时间运行操作而不提供界面的应用组件。服务可由其他应用组件启动，而且即使用户切换到其他应用，服务仍将在后台继续运行。此外，组件可通过绑定到服务与之进行交互，甚至是执行进程间通信 (IPC)。

service拥有3种不同类型：

前台 前台服务执行一些用户能注意到的操作。

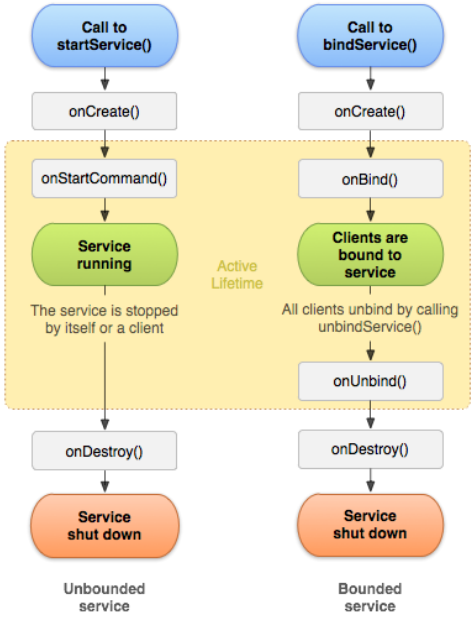
后台 后台服务执行用户不会直接注意到的操作。

绑定 当应用组件通过调用 bindService() 绑定到服务时，服务即处于绑定状态。

Service有三种状态：（启动状态）这个时候Service内部可以做一些后台计算，并且不需要和外界有直接的交互，要去实现onStartCommand方法；（绑定状态）这个时候Service内部同样可以进行后台计算，但是处于这种状态时外界可以很方便地和Service组件

进行通信，要求实现onBind方法。（混合状态）以上两种的结合。

了解服务的生命周期




## 验证实验一：了解启动状态

1. 掌握服务的创建方法

模仿example下的Exp1Service创建一个服务（创建服务,右击exp->New->Service->Service,输入文件名，点击Finish）

查看AndroidManifest.xml文件，观察的具有service标签的元素。

android:name 属性是唯一必需的属性，用于指定服务的类名。

android:enabled 系统是否可实例化服务 —"true"表示可以，"false"表示不可以。默认值为"true"。

android:exported 其他应用的组件是否能调用服务或与之交互 —"true"表示可以，"false"表示不可以。当该值为"false"时，只有同一个应用或具有相同用户 ID 的应用的组件可以启动服务或绑定到服务。默认值取决于服务是否包含 Intent 过滤器。

参考：*https://developer.android.com/guide/topics/manifest/service-element#desc*

2. 启动与停止服务

其他组件可以通过startService方法可以用来启动服务，这会调用服务的onStartCommand方法。

在example下提供了本实验源码，分别是Exp1Service和Exp1Activity两个文件。

在模拟器中打开这个应用，多次点击"启动服务"按钮和"停止服务"按钮，使用Logcat查看onCreate、 onStartCommand、onDestory方法执行情况。（过滤条件：Level为Info，关键词为Exp1）

onCreate() 首次创建服务时，系统会（在调用 onStartCommand() 或 onBind() 之前）调用此方法来执行一次性设置程序。如果服务已在运行，则不会调用此方法。

onStartCommand() 当另一个组件（如 Activity）请求启动服务时，系统会通过调用 startService() 来调用此方法。如果您只想提供绑定，则无需实现此方法。

onDestroy()当不再使用服务且准备将其销毁时，系统会调用此方法。服务应通过实现此方法来清理任何资源，如线程、注册的侦听器、接收器等。这是服务接收的最后一个调用。

参考：*https://developer.android.com/guide/components/services#Basics*

3. 通过修改Exp1Service中的execTime来模拟服务执行，并理解服务是运行在MainThread里的，服务中onStartCommand执行时间过长会引发ANR，同时讨论使用多线程的必要性。

4. 通过修改execAutoStop来模拟服务执行完自动停止的情况，提交Logcat上显示的启动情况。（过滤条件：Level为Info，关键词为Exp1）

本验证实验提交内容：

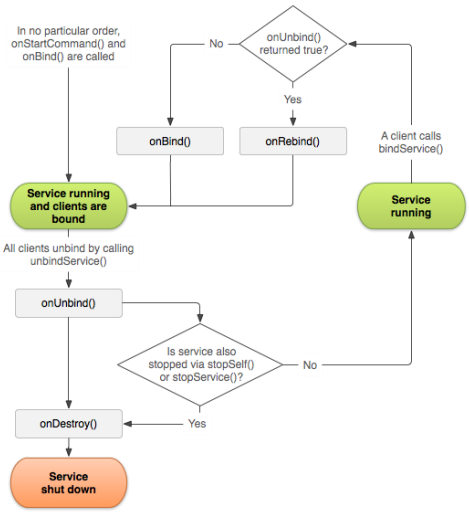
1. 截取Logcat、关于这些Logcat的分析。
2. 区分服务与多线程

## 验证实验二：了解绑定状态

1. 启动服务以后，出现了新的需求：让Activity和Service更加紧密，使得Activity可以指挥Service。这需要Activity和Service之间能够进行通讯。

绑定服务允许应用组件通过调用 bindService() 与其绑定，从而创建长期连接。服务只用于与其绑定的应用组件，因此若没有组件与该服务绑定，则系统会销毁该服务。

了解绑定状态的生命周期




2. 绑定服务

在example下提供了相关源码，分别是Exp2Service和Exp2Activity两个文件。

Binder：从Android应用层而言，Binder是客户端和服务器端进行通讯的媒介，在Android开发中，Binder主要用在Service中。

ServiceConnection：调用时，它必须提供 ServiceConnection 的实现，后者会监控与服务的连接。bindService() 的返回值表明所请求的服务是否存在，以及是否允许客户端访问该服务。当创建客户端与服务之间的连接时，Android 系统会调用 ServiceConnection 上的 onServiceConnected()。onServiceConnected() 方法包含 IBinder 参数，客户端随后会使用该参数与绑定服务进行通信。

可以同时将多个客户端连接到服务。但是，系统会缓存 IBinder 服务通信通道。换言之，只有在第一个客户端绑定服务时，系统才会调用服务的 onBind() 方法来生成 IBinder。然后，系统会将同一 IBinder 传递至绑定到相同服务的所有其他客户端，无需再次调用 onBind()。

点击"连接一"下的"绑定服务"按钮和"链接二"下的"绑定按钮", 通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp2）

3. 使用服务

连接一中获取了Exp2Service对象，可以调用Exp2Service中的方法。

连接二中获取了Exp2Service.Exp2Binder对象，可以调用Exp2Service.Exp2Binder中的方法。

点击"连接一"下的"测试活动和服务间的通讯"按钮和"链接二"下的"测试活动和服务间的通讯", 通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp2）

参考：*https://developer.android.com/guide/components/bound-services#Binder*

4. 解绑服务

了解unbind是bind的逆操作，主要是清理bind相关对象，当一个绑定模式的服务 全部绑定被取消以后，自动调用onDestroy方法。了解unbind一个未bind的ServiceConnection会导致程序崩溃。

了解onServiceDisconnected并不是onServiceConnected的逆操作。unbind方法并不会回调onServiceDisconnected.所以，在unbind以后，对于客户端进程，引用仍然存在，只是不再接收死亡通知了。根据官方文档：当与服务的连接意外中断（例如服务崩溃或被终止）时，Android 系统会调用该方法。当客户端取消绑定时，系统*不会*调用该方法。

了解unbindService后仍可继续使用缓存的binder。这是由于强引用造成的系统不能GC。

点击"连接一"下的"解绑服务"按钮和"链接二"下的"解绑服务", 通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp2）

然后点击"连接一"下的"测试活动和服务间的通讯"按钮, 通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp2）

本验证实验提交内容：

1. 截取Logcat、关于这些Logcat的分析。

参考：*https://zhuanlan.zhihu.com/p/36892395*

## 验证实验三：了解前台服务

1. 前台服务是用户主动意识到的一种服务，因此在内存不足时，系统也不会考虑将其终止。前台服务必须为状态栏提供通知，将其放在运行中的标题下方。这意味着除非将服务停止或从前台移除，否则不能清除该通知。

如要从前台移除服务，请调用 stopForeground()。此方法采用布尔值，指示是否需同时移除状态栏通知。此方法不会停止服务。但是，如果您在服务仍运行于前台时将其停止，则通知也会随之移除。

了解服务启动采用Intent作为参数，可以进行参数传递。

了解前台服务通过调用startForceground来实现的，该方法要求两个参数：唯一标识通知的整型数和用于状态栏的 Notification,了解到Android 8 以后需要借助channel才能使用Notification。了解到Android 9以后，使用前台服务必须申请权限：FOREGROUND_SERVICE。

2. 使用前台服务

在example下提供了相关源码，分别是Exp3Service和Exp3Activity两个文件。

分别点击"开启后台服务"、"开启前台服务"、"停止前台服务"、"停止该服务"，观察通知栏变化，通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp3）

本次小实验提交内容： 截取Logcat、 关于这些Logcat的分析

## 验证实验四：认识多线程

1. 复习线程的基本用法，包括Thread、Runnable。

了解到Android的UI和大多数GUI库一样是线程不安全的，Android必须在UI线程中更新UI。

了解Android异步消息处理机制，了解Message、Handle、MessageQueue、Looper

Message是在线程中传递的消息，可以在内部携带少量的信息，用于不同线程之间进行交换数据

Handle用于发送和处理消息，使用Handle的sendMessage，经过一系列辗转，最终传递到HandleMessage

MessageQueue存放所有通过Handle发送的消息，这部分消息会一直存放在消息队列中，等待被处理，每个线程只会有一个MessageQueue对象

Looper是每个线程中的MessageQueue管家，调用Looper的loop()方法后，就会进入无线循环，发现MessageQueue存在消息，取出并且传递到handleMessage

了解Handle直接定义为Activity会由于内部类隐式持有外部类指针而导致Activity无法销毁进而导致内存泄漏。

了解RunOnUiThread方法，该方法接收一个实现了Runnable接口的类，可以切换到Ui线程。

2. 在example下提供了相关源码，分别是Exp4Service和Exp4Activity两个文件。

点击“非UI线程改变文本”、“异步改变文本，采用Handle”、“异步改变文本，采用runOnUiThread”，观察文本框变化，通过Logcat查看各个方法的执行情况。（过滤条件：Level为Info，关键词为Exp4）

参考：https://www.jianshu.com/p/58c999d3ada7

本次小实验提交内容： 截取Logcat、 关于这些Logcat的分析

## 验证实验五：认识AsyncTask

认识AsyncTask：AsyncTask基于异步消息处理机制

了解AsyncTask的3个泛型参数：params、progress、result；经常重写的方法：

onPreExecute、doInBackgroup、onProgressUpdate、onPostExecute

1. Params：在执行AsyncTask时需要传入的参数、可用于后台任务中使用

2. Progress：后台任务执行时，如果需要在界面上显示当前的进度，则使用这里指定的泛型作为进度单位

3. Result：当任务执行完毕以后，如果需要对任务进行返回，则使用这里指定的泛型作为返回类型

4. onPreExecute()：在后台任务开始执行前调用，用于界面上一些初始化工作

5. doInBackgroud(Params…)：在子线程中执行，可以通过调用publishProgress方法反馈当前任务进度

6. onProgressUpdate(Progress…)：当后台任务调用了publishProgress后，onPregressUpdate很快被调用，该方法可以进行UI操作

7. onPostExecute(Result)：当后台执行完毕并通过return语句返回时，该方法很快被调用。

了解由于AsyncTask在创建的时候会持有Activity的引用而造成内存泄漏的问题

实验内容：

1. 通过Logcat工具了解android的AsyncTask（示例文件：example下的Exp5Activity、Exp5Service），分析Logcat，提交Logcat显示的信息（过滤条件：Level为Info，关键词为Exp5），分析原因

## 自选实验

自选实验可以从以下两个选项中选择一个，也可以自选题目

1. 下载服务

   编写一个具有下载功能的安卓应用。建议2-4人组队进行。

2. 音乐服务

   编写一个具有音乐播放功能的安卓应用。建议2-4人组队进行。

拓展知识：了解AIDL，了解跨进程服务绑定，了解RxJava