第十章

后台默默的劳动者--探究服务

主讲：王海

Service简介

- Service组件表示一种服务，专门用于执行一些持续性的、耗时长的并且无需与用户界面交互的操作
- Service的运行是不可见的，通常用于执行一些无需用户交互，并需要持续运行的任务
- Service拥有独立的生命周期
- Service没有界面（最多只能显示一个通知），当Service所对应的应用程序界面不可见时，Service仍运行于应用程序主线程中
- Android系统中提供了大量可以直接调用的系统Service，例如播放音乐、震动、闹钟、通知栏消息等

Service分类

- 按照运行的进程不同，可以将Service分为：
  - ✓ 本地（Local）Service
  - ✓ 远程（Remote）Service
- 按照运行的形式分为：
  - ✓ 前台Service
  - ✓ 后台Service
- 按照使用Service的方式可以分为：
  - ✓ 启动（Start）方式Service
  - ✓ 绑定（Bind）方式Service
  - ✓ 混合方式Service

创建Service的步骤

①通过继承Service的方式来定义一个Service的子类

②在应用程序的AndroidManifest.xml中配置Service组件

编写Service类

- 【语法】`public abstract IBinder onBind(Intent intent);`

- 【案例8- 1】MyService1.java

```
// 一个空的Service示例
public class MyService1 extends
Service {
    @Override
    public IBinder
onBind(Intent intent) {
        return null;
    }
}
```

配置Service

- 在AndroidManifest.xml中，每个Service组件都需要在<application>元素的一个<service>子元素中进行配置

```
<service
android:name="com.example.zhaokl.chapter08.MyService1" />
```

启动Service

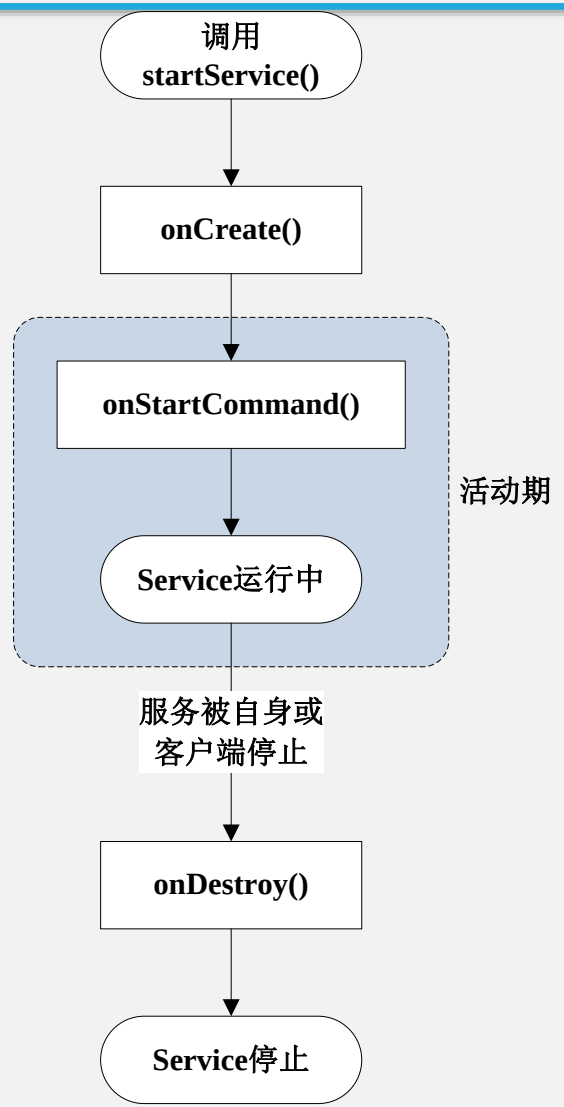
- 启动Service有Start和Bind两种方式

```
Intent intent = new Intent(this, MyService1.class);
startService(intent);
```

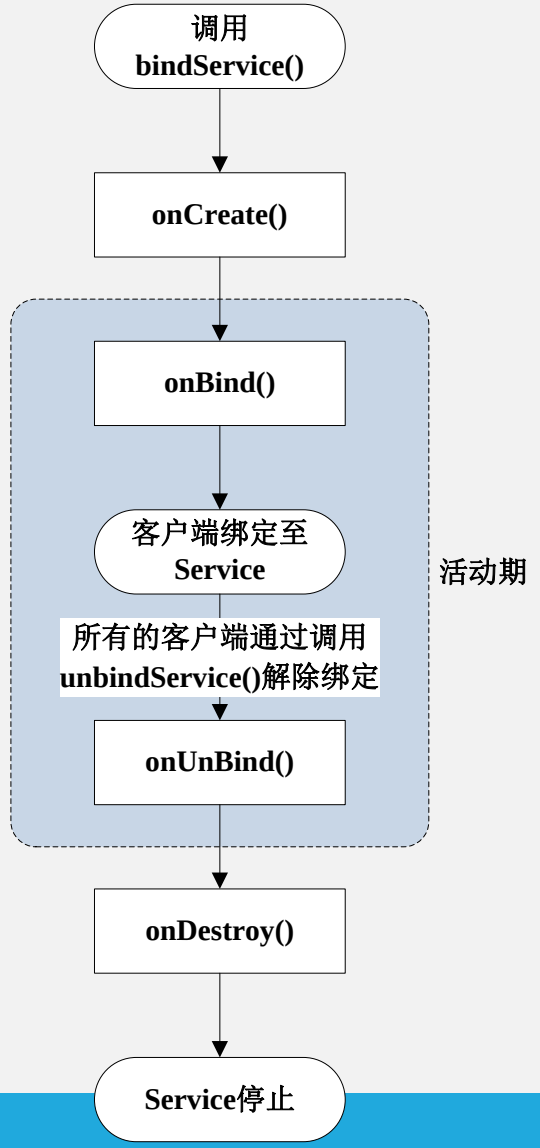
Service生命周期回调方法

方法	功能描述
onCreate()	用于创建Service组件
onStartCommand(Intent intent, int flags, int started)	通过Start方式启动Service时调用
onBind(Intent intent)	通过Bind方式启动Service
onUnbind(Intent intent)	通过Bind方式取消Service绑定
onRebind(Intent intent)	通过Bind方式重新绑定Service
onDestroy()	用于销毁Service

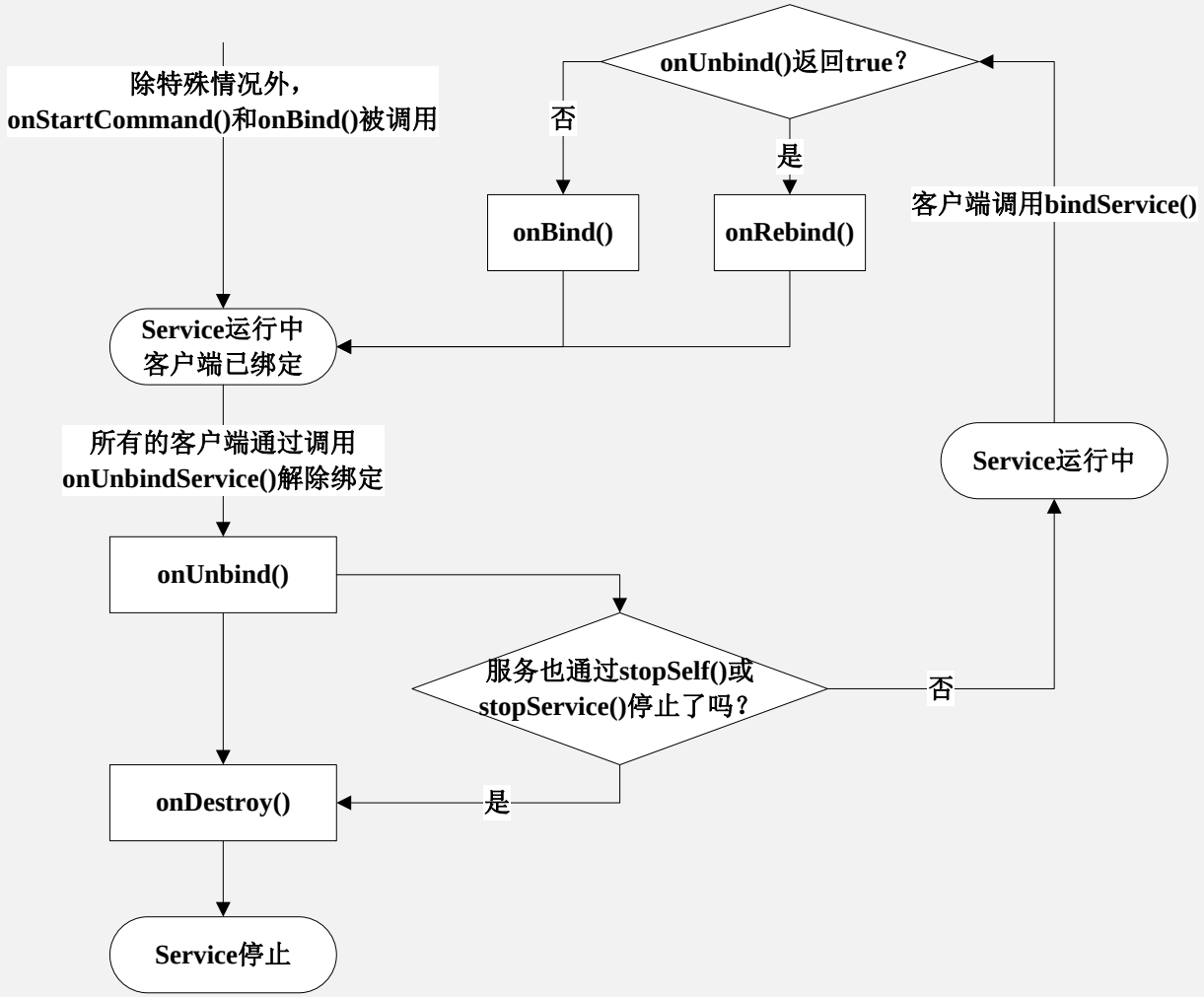
Start方式启动Service



Bind方式启动Service



混合方式的Service



进程类型

- 前台进程（Foreground Process）
- 可见进程（Visible Process）
- 服务进程（Service Process）
- 后台进程（Background Process）

前台Service

- Service启动后，其所在进程默认是服务进程，优先级并不高
- 通过Service的startForeground()方法将Service改为前台进程
- 【语法】
```
public final void startForeground(int id,
Notification notification)
```
  - ✓ 参数id：通知的id
  - ✓ 参数notification：需要显示的通知

– 调用stopForeground()方法取消其前台状态

– 【语法】

```
public final void stopForeground(boolean
removeNotification)
```

Service中执行耗时任务

- 针对在Service中执行耗时任务，Android还专门提供了一种特殊的Service：IntentService
- 抽象类android.app.IntentService是Service的子类
- IntentService会自动开始一个新线程来执行任务，并在任务执行完毕后停止Service
- 使用IntentService非常简单，只需继承IntentService并重写onHandleIntent()方法即可

远程Service

- 远程Service允许被另一个进程中的组件访问
- AIDL（Android Interface Definition Language，Android接口定义语言）是Android提供的一种专门用于描述进程间通信接口的语言
- 使用AIDL可以简化在进程间交换数据的代码，使客户端可以像本地Service那样直接绑定远程Service

系统自带Service

- Android提供了许多系统级别的Service，通过这些服务应用程序可以方便的调用系统功能
- 系统服务都是通过Context.getSystemService(String serviceName)方法获取

服务对象	Context中对应的服务名称常量	功能
AccessibilityManager	ACCESSIBILITY_SERVICE	通过已注册的事件监听器将UI事件反馈给用户
AccountManager	ACCOUNT_SERVICE	账户服务
ActivityManager	ACTIVITY_SERVICE	管理Activity、Service等各种组件
AlarmManager	ALARM_SERVICE	闹钟服务
AppOpsManager	APP_OPS_SERVICE	在设备操作时跟踪应用
AudioManager	AUDIO_SERVICE	音频服务
BluetoothAdapter	BLUETOOTH_SERVICE	蓝牙服务
ClipboardManager	CLIPBOARD_SERVICE	剪切板服务

服务对象	Context中对应的服务名称常量	功能
ConnectivityManager	CONNECTIVITY_SERVICE	网络连接服务
ConsumerIrManager	CONSUMER_IR_SERVICE	红外信号服务
DevicePolicyManager	DEVICE_POLICY_SERVICE	设备监听服务
DisplayManager	DISPLAY_SERVICE	显示设备管理
DownloadManager	DOWNLOAD_SERVICE	针对HTTP的下载服务
DropBoxManager	DROPBOX_SERVICE	获取 DropBoxManager 实例以记录诊断日志
InputMethodManager	INPUT_METHOD_SERVICE	输入法的管理服务程序
InputManager	INPUT_SERVICE	输入设备管理
NotificationManager	KEYGUARD_SERVICE	键盘锁服务
LayoutInflater	LAYOUT_INFLATER_SERVICE	根据XML生成布局的服务
LocationManager	LOCATION_SERVICE	GPS定位服务等
NfcManager	NFC_SERVICE	NFC服务
NotificationManager	NOTIFICATION_SERVICE	通知服务
PowerManager	POWER_SERVICE	电源服务
PrintManager	PRINT_SERVICE	打印服务
SearchManager	SEARCH_SERVICE	搜索服务
SensorManager	SENSOR_SERVICE	传感器服务
StorageManager	STORAGE_SERVICE	系统存储服务
TelephonyManager	TELEPHONY_SERVICE	电话服务
TextServicesManager	TEXT_SERVICES_MANAGER_SERVICE	文字服务，如拼写检查等
UiModeManager	UI_MODE_SERVICE	界面模式服务，如夜间模式、驾车模式等
UsbManager	USB_SERVICE	USB管理服务
UserManager	USER_SERVICE	用户管理服务
Vibrator	VIBRATOR_SERVICE	振动器服务
WallpaperService	WALLPAPER_SERVICE	壁纸服务
WifiP2pManager	WIFI_P2P_SERVICE	WIFI-P2P连接服务
WifiManager	WIFI_SERVICE	WIFI服务
WindowManager	WINDOW_SERVICE	系统窗口服务

系统自带Service

- NotificationManager用于在界面顶部的通知栏显示消息，以一种标准的方式向用户显示提示信息
- DownloadManager提供了一种标准简洁的HTTP下载解决方案

本章总结

- 按照运行的进程不同，可以将Service分为本地Service和远程Service
- 按照运行的形式分为前台Service和后台Service
- 按照使用Service的方式可以分为启动方式Service、绑定方式Service、和混合式Service
- Service组件需要通过Context对象启动，有两种启动方式，Start方式和Bind绑定方式，分别对应于Context的startService()和bindService()方法
- 无论是Start还是Bind方式启动Service，都会经历onCreate()和onDestroy()方法；如果是Start方式启动，在启动时会调用onStartCommand()方法；如果是Bind方式启动，在启动时会调用onBind()方法，取消绑定时会调用onUnbind()方法，重新绑定时会调用onRebind()方法
- start方式启动的Service必须自己管理生命周期，并会一直运行下去，除非Service调用自身的stopSelf()方法，或其他组件对该Service调用stopService()方法

本章总结

- bind方式启动的Service会和启动它的组件关联在一起并可以进行通信
- Service启动后，其所在进程默认是服务进程，优先级并不高，如果是非常重要的Service，可以通过调用Service的startForeground()方法将其改为前台进程
- IntentService是Service的子类，其内部会自动开始一个新线程执行任务，并在任务执行完毕后停止Service
- 远程Service是指运行于独立的进程中，并为其他进程提供服务的Service。调用远程Service时需要使用AIDL
- Android提供了许多系统级的Service，利用这些服务，应用程序可以方便的调用系统功能，通过Context.getSystemService(String serviceName)方法可以获取这些服务对象