# 1. 实验目的

了解Android服务(Service)的构成;

熟练掌握启动与绑定两种不同的调用服务的方法;

熟练掌握Android本地服务(Local Service)的使用方法;

熟练掌握AIDL接口的使用方法;

熟练掌握Android远程服务(Remote Service)的使用方法

# **2. 实验内容**

在线下载Android开发工具，安装工具并搭建开发环境

创建一个Android开发的简单项目

# **3. 实验步骤与结果**

1) 创建服务时，必须实现Service的onBind()方法

2）Service有两种启动方式，分别是：

1）startService()

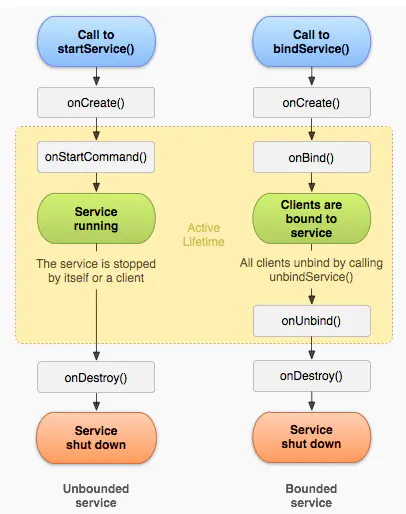
2）bindService()

3) Service分类：

1）本地服务 Local Service，它是依赖与主线程运行的，这样某种程度可以节省一些资源， 既然依附于主线程，那就不需要IPC，主线程被kill，那此service也就终止。

2）远程服务 Remote Service，它是一个单独的进程运行，可以在AndroidManifest清单文 件中通过android:porcess来设定。那如果存在IPC，则就需要AIDL，但是其存在灵活性， 可以为多个进程服务，会占用必定资源。

4）Service的生命周期：



第一种方式：经过startService启动Service

经过startService启动后，service会一直无限期运行下去，只有外部调用了stopService()或stopSelf()方法时，该Service才会中止运行并销毁。

要建立一个这样的Service，要让该类继承Service类，而后重写如下方法：

onCreate()

1.若是service没被建立过，调用startService()后会执行onCreate()回调；

2.若是service已处于运行中，调用startService()不会执行onCreate()方法。

也就是说，onCreate()只会在第一次建立service时候调用，屡次执行startService()不会重复调用onCreate()，此方法适合完成一些初始化工做。

onStartCommand()

若是屡次执行了Context的startService()方法，那么Service的onStartCommand()方法也会相应的屡次调用。onStartCommand()方法很重要，咱们在该方法中根据传入的Intent参数进行实际的操做，好比会在此处建立一个线程用于下载数据或播放音乐等。

onBind()

Service中的onBind()方法是抽象方法，Service类自己就是抽象类，因此onBind()方法是必须重写的，即便咱们用不到。

onDestory()

在销毁的时候会执行Service该方法。

这几个方法都是回调方法，且在主线程中执行，由android操做系统在合适的时机调用。

须要注意，项目中的每个Service都必须在AndroidManifest.xml中注册才行，因此还须要编辑AndroidManifest.xml文件，代码以下所示：

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="kylec.me.servicedemo"

android:versionCode="1"

android:versionName="1.0" >

<uses-sdk

android:minSdkVersion="21"

android:targetSdkVersion="31" />

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<service android:name="kylec.me.servicedemo.MyService" >

</service>

</application>

</manifest>

Service代码：

public class MyService extends Service{

@Override

public void onCreate() {

Log.i("Kathy","onCreate - Thread ID = " + Thread.currentThread().getId());

super.onCreate();

}

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

Log.i("Kathy", "onStartCommand - startId = " + startId + ", Thread ID = " + Thread.currentThread().getId());

return super.onStartCommand(intent, flags, startId);

}

@Nullable

@Override

public IBinder onBind(Intent intent) {

Log.i("Kathy", "onBind - Thread ID = " + Thread.currentThread().getId());

return null;

}

@Override

public void onDestroy() {

Log.i("Kathy", "onDestroy - Thread ID = " + Thread.currentThread().getId());

super.onDestroy();

}

}

启动Service：

Intent service= new Intent(this, MyService.class);

startService(service);

终止Service：

stopService(service);

第二种方式：经过bindService启动Service

bindService启动服务特色：

1. bindService启动的服务和调用者之间是典型的client-server模式。调用者是client，service则 是server端。service只有一个，但绑定到service上面的client能够有一个或不少个。这里所提 到的client指的是组件，好比某个Activity。

2. client能够经过IBinder接口获取Service实例，从而实如今client端直接调用Service中的方 法以实现灵活交互，这在经过startService方法启动中是没法实现的。

3. bindService启动服务的生命周期与其绑定的client息息相关。当client销毁时，client会自动 与Service解除绑定。固然，client也能够明确调用Context的unbindService()方法与Service解 除绑定。当没有任何client与Service绑定时，Service会自行销毁。

Service代码如下：

public class TestTwoService extends Service{

//client 能够经过Binder获取Service实例

public class MyBinder extends Binder {

public TestTwoService getService() {

return TestTwoService.this;

}

}

//经过binder实现调用者client与Service之间的通讯

private MyBinder binder = new MyBinder();

private final Random generator = new Random();

@Override

public void onCreate() {

Log.i("Kathy","TestTwoService - onCreate - Thread = " + Thread.currentThread().getName());

super.onCreate();

}

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

Log.i("Kathy", "TestTwoService - onStartCommand - startId = " + startId + ", Thread = " + Thread.currentThread().getName());

return START\_NOT\_STICKY;

}

@Nullable

@Override

public IBinder onBind(Intent intent) {

Log.i("Kathy", "TestTwoService - onBind - Thread = " + Thread.currentThread().getName());

return binder;

}

@Override

public boolean onUnbind(Intent intent) {

Log.i("Kathy", "TestTwoService - onUnbind - from = " + intent.getStringExtra("from"));

return false;

}

@Override

public void onDestroy() {

Log.i("Kathy", "TestTwoService - onDestroy - Thread = " + Thread.currentThread().getName());

super.onDestroy();

}

//getRandomNumber是Service暴露出去供client调用的公共方法

public int getRandomNumber() {

return generator.nextInt();

}

}

启动Service：

Intent service= new Intent(this, TestTwoService.class);

bindService(service, conn, BIND\_AUTO\_CREATE);

停止Service：

unbindService(conn)

# **4. 分析与讨论**

通过本次实验我了解了Android的Service，Service是一个后台运行的组件，用来执行长时间运行且不需要用户交互的任务。这次实验真的是让我受益匪浅，学到了学到了。