# 《移动终端开发技术》

# 电子教案

## 第三单元　服务的更多技巧

所属专业（教研室）：计算机软件技术

制定人：陈媛媛

合作人：

制定时间：2018 年 2 月

日照职业技术学院

<table>
<tr><td>单元标题</td><td colspan="2">服务的更多技巧</td><td>单元教学学时</td><td>2 课时</td></tr>
<tr><td></td><td></td><td>在整体设计中的位置</td><td>第 25 次</td></tr>
<tr><td>授课班级</td><td></td><td>上课地点</td><td>一体化教室</td></tr>
<tr><td>上课时间</td><td colspan="3">周　　月　　日第　　节</td></tr>
<tr><td rowspan="2">教学<br>目标</td><td>能力目标</td><td>知识目标</td><td>素质目标</td></tr>
<tr><td>能够熟练运用前台服务和 IntentService。</td><td>1、掌握前台服务的用法；<br>2、掌握 IntentService 的用法。</td><td>1、养成积极主动学习意识；<br>2、养成勤于动手的习惯。</td></tr>
<tr><td>教学重点、<br>难点</td><td colspan="3">教学重点：前台服务的用法<br>教学难点：IntentService 的用法</td></tr>
<tr><td>教学<br>方法</td><td colspan="3">采用反转课堂教学模式，课前学生学习微课了解知识点，课上采用教师引导、演示，学生分组练习、讨论等教学方法。<br>运用多媒体、AndroidStudio 开发环境、实训助手、教学平台等辅助授课。</td></tr>
<tr><td>课前需掌握的知识点</td><td colspan="3">Notification notification=new NotificationCompat.Builder(this)<br>.setContentTitle("this is content title")//用于指定通知的标题内容<br>.setContentText("this is content text")//用于指定通知的正文内容<br>.setWhen(System.currentTimeMillis())//用于指定通知被创建的时间 .setSmallIcon(R.mipmap.ic_launcher)//用于设置通知的小图标<br>.setLargeIcon(BitmapFactory.decodeResource(getResources(),R.mipmap.ic_launcher))<br>.build();</td></tr>
<tr><td>教学任务<br>分解</td><td colspan="3">任务一、使用前台服务<br>任务二、使用 IntentService</td></tr>
<tr><td>教学<br>总结</td><td colspan="3"></td></tr>
</table>

教学过程

## 一、知识回顾

1、对上节课留的作业进行答疑。

2、回顾总结上节课的内容，引出本节课主题

服务几乎都是在后台运行的。但是服务的系统优先级还是比较低的，当系统出现内存不足的情况时，就有可能会回收掉正在后台运行的服务。如果你希望服务可以一直保持运行状态，而不会由于系统内存不足的原因导致被回收，就可以考虑使用前台服务。

## 二、课前复习

通知（Notification）的基本用法。

## 三、知识讲解

### 任务一、使用前台服务

前台服务和普通服务最大的区别就在于，它会一直有一个正在运行的图标在系统的状态栏显示，下拉状态栏后可以看到更加详细的信息，非常类似于通知的效果。有些项目由于特殊的需求会要求必须使用前台服务，比如说墨迹天气，它的服务在后台更新天气数据的同时，还会在系统状态栏一直显示当前的天气信息，如图 1 所示。


图 1 前台服务

修改 MyService 中 的代码，如下所示：

```
public class MyService extends Service {
……
@Override
public void onCreate() {
super.onCreate();
Log.d("MyService", "onCreate executed");
 Intent intent = new Intent(this, MainActivity.class);
```

```
// PendingIntent,待确定的意图,等待的意图。
PendingIntent pi = PendingIntent.getActivity(this, 0, notificationIntent, 0);
// 构建 Notification 实例
Notification notification = new NotificationCompat.Builder(this)
.setContentTitle(“this is content title”)
.setContentText(“this is content text”)
.setWhen(System.currentTimeMillis())
.setSmallIcon(R.mipmap.ic_launcher)
.setLargeIcon(BitmapFactory.decodeResource(getResources(),R.mipmap.ic_laucher))
.setContentIntent(pi)
.build();
//调用 startForeground()方法后就会让 MyService 变成一个前台服务
startForeground(1,notification);
}
……
}
```

可以看到，这里只是修改了 onCreate()方法中的代码，这是我们在上一章中学习的创建通知的方法。只不过这次在构建出 Notification 对 象后并没有使用 NotificationManager 来将通知显示出来，而是调用了 startForeground()方法。这个方法接收两个参数，第一个参数是通知的 id，类似于 notify()方法的第一个参数，第二个参数则是构建出的 Notification 对象。调用 startForeground()方法后就会让 MyService 变成一个前台服务，并在系统状态栏显示出来。

现在重新运行一下程序，并点击 Start Service 或 Bind Service 按钮，MyService 就会以前台服务的模式启动了，并且在系统状态栏会显示一个通知图标，下拉状态栏后可以看到该通知的详细内容，如图 2 所示。

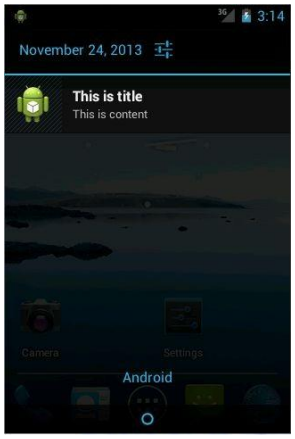


图 2 前台服务运行效果

**任务二、使用 IntentService**

在本章一开始的时候我们就已经知道，服务中的代码都是默认运行在主线程当中的，如果直接在服务里去处理一些耗时的逻辑，就很容易出现 ANR（Application Not Responding）的情况。所以这个时候就需要用到 Android 多线程编程的技术了，我们应该在服务的每个具体的方法里开启一个子线程，然后在这里去处理那些耗时的逻辑。因此，一个比较标准的服务就可以写成如下形式：

```
public class MyService extends Service {
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
new Thread(new Runnable() {
@Override
public void run() {
// 处理具体的逻辑
}
}).start();
return super.onStartCommand(intent, flags, startId);
 }
}
```

但是，这种服务一旦启动之后，就会一直处于运行状态，必须调用 stopService()或者 stopSelf()方法才能让服务停止下来。所以，如果想要实现让一个服务在执行完毕后自动停止的功能，就可以这样写：

```
public class MyService extends Service {
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
new Thread(new Runnable() {
@Override
public void run() {
// 处理具体的逻辑
stopSelf();
 }
 }).start();
 return super.onStartCommand(intent, flags, startId);
 }
}
```

虽说这种写法并不复杂，但是总会有一些程序员忘记开启线程，或者忘记调用 stopSelf() 方法。为了可以简单地创建一个异步的、会自动停止的服务，Android 专门提供了一个 IntentService 类，这个类就很好地解决了前面所提到的两种尴尬，下面我们就来看一下它的用法。

新建一个 MyIntentService 类继承自 IntentService，代码如下所示：

```
public class MyIntentService extends IntentService {
 public MyIntentService() {
     super("MyIntentService"); // 调用父类的有参构造函数
     }
 @Override
     protected void onHandleIntent(Intent intent) {
     // 打印当前线程的 id
     Log.d("MyIntentService", "Thread id is " + Thread.currentThread(). getId
     ());
     }
 @Override
     public void onDestroy() {
     super.onDestroy();
     Log.d("MyIntentService", "onDestroy executed");
     }
}
```

这里首先是要提供一个无参的构造函数，并且必须在其内部调用父类的有参构造函数。

然后要在子类中去实现 onHandleIntent()这个抽象方法，在这个方法中可以去处理一些具体的 逻辑，而且不用担心 ANR 的问题，因为这个方法已经是在子线程中运行的了。这里为了证 实一下，我们在 onHandleIntent()方法中打印了当前线程的 id。另外根据 IntentService 的特性， 这个服务在运行结束后应该是会自动停止的，所以我们又重写了 onDestroy()方法，在这里也打印了一行日志，以证实服务是不是停止掉了。

接下来修改 activity_main.xml 中的代码，加入一个用于启动 MyIntentService 这个服务的按钮，如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" androi
d:layout_width="match_parent" android:layout_height="match_parent"
android:orientation="vertical" >
……
  <Button android:id="@+id/start_intent_service" android:layout_width="match_parent
" android:layout_height="wrap_content" android:text="Start IntentService" />
</LinearLayout>
```

然后修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity implements OnClickListener {
……
```

```
 private Button startIntentService;
  @Override
  protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstance
State); setContentView(R.layout.activity_main);
……
  startIntentService = (Button) findViewById(R.id.start_intent_service);
  startIntentService.setOnClickListener(this);
}
  @Override
  public void onClick(View v) {
  switch (v.getId()) {
  ……
  case R.id.start_intent_service:
// 打印主线程的 id
  Log.d("MainActivity", "Thread id is " + Thread.currentThread().
  getId());
   Intent intentService = new Intent(this, MyIntentService.class);
   startService(intentService);
   break;
   default:
   break;
   } }
   }
```

可以看到，我们在 Start IntentService 按钮的点击事件里面去启动 MyIntentService 这个服务，并在这里打印了一下主线程的 id，稍后用于和 IntentService 进行比对。你会发现，其实 IntentService 的用法和普通的服务没什么两样。最后仍然不要忘记，服务都是需要在 AndroidManifest.xml 里注册的，如下所示：

```
    <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="c
   om.example.servicetest"
   android:versionCode="1"
   android:versionName="1.0" >
   ……
   <application android:allowBackup="true" android:icon="@drawable/ic_launcher" andr
   oid:label="@string/app_name" android:theme="@style/AppTheme" >
   ……
   <service android:name=".MyIntentService"></service>
   </application>
```

</manifest>

现在重新运行一下程序，界面如图 3 所示。

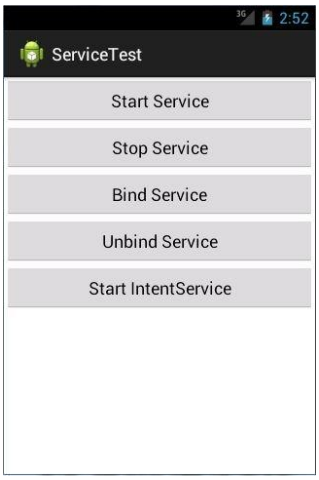


图 3 IntentService

点击 Start IntentService 按钮后，观察 LogCat 中的打印日志，如图 4 所示。

| Tag | Text |
| --- | --- |
| MainActivity | Thread id is 1 |
| MyIntentService | Thread id is 97 |
| MyIntentService | onDestroy executed |

图 4 LogCat 中的打印日志

可以看到，不仅 MyIntentService 和 MainActivity 所在的线程 id 不一样，而且 onDestroy() 方法也得到了执行，说明 MyIntentService 在运行完毕后确实自动停止了。集开启线程和自动停止于一身，IntentService 还是博得了不少程序员的喜爱。

## 四、演示作品

各小组演示汇报作品，教师与其他小组总结评价作品，各小组课后完善作品并提交到教学平台。

## 五、知识巩固

总结知识点，使用教学平台中的随堂练习题巩固本所学知识。

| | |
| --- | --- |
| **拓展作业** | 1、 在项目中运用前台服务；<br>2、 了解 IntentService 的更多技巧 |
| **教学后记** | |