移动应用开发课程作业
背景音乐 service 的实现

指导老师：曹阳

2017 年 12 月 18 日

华南师范大学计算机学院
计算机科学与技术（软件技术应用方向）
2015 级 5 班
王子彦
20142100012
ziyan.wang@m.scnu.edu.cn

软件名称	MyDiaryBook
完成人	王子彦
学号	20142100012
完成时间	2017 年 12 月 18 日

一、 软件内容简介

本次作业在上次日记本 app 的基础上通过 bindService 和 AsyncTask 多线程技术实现了播放背景音乐功能，通过权限申请和 SD 卡状态检查确保 app 的正常运行，并通过 ContentResolver 获取系统中所有的音乐。

您可在此处查阅本项目的所有源码。

二、界面设计

由于本次 app 的重点在于后台 service 的业务逻辑，因此界面较简单，只有一个控制菜单。用户在点击播放音乐按钮后，将会自动扫描并播放设备上的音乐。

图 2-1 用户界面展示图

三、代码设计

由于代码量较大，下面选择几项重要的部分展示。如有兴趣，请在 Android Studio 中 clone 开发者的 Github 代码仓库，打开项目文件进行详细查阅。

3.1 背景音乐播放服务

为了更好的实现 UI 线程与 background service 之间的通信，本次设计中使用了 bindService() 和 unbindService()来启动和停止服务，图 3-2 和 3-3 是背景音乐播放服务的主要代码实现。

在此处并没有用到多线程（下文扫描音乐用到了），service 和 activity 实际上在同一个线程中执行。之所以这样设计，是因为 app 的业务逻辑更适合用 Service 而不是 Thread。具体请参见 Stack Overflow 上对服务和线程的解析：[Service vs Thread in Android]

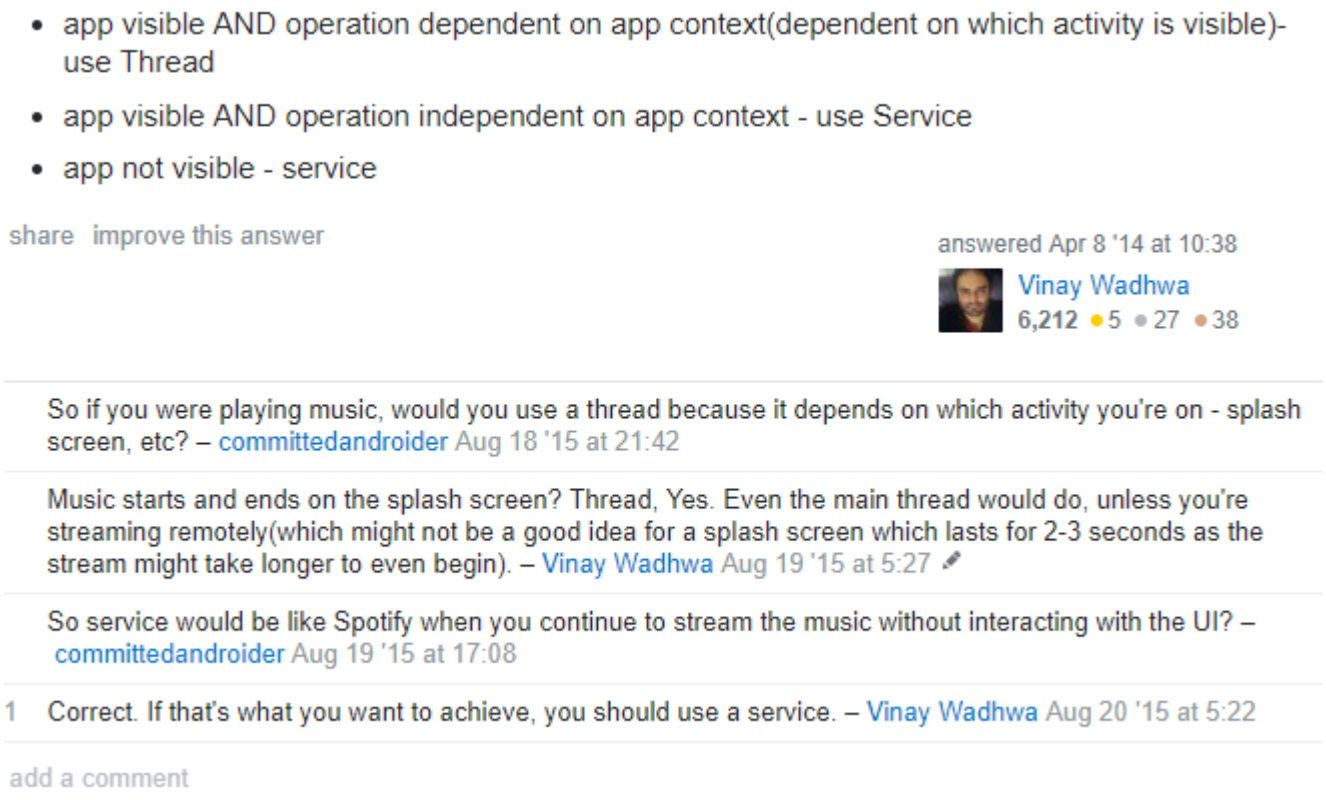

图 3-1 关于 Thread 和 Service 适用场景的解析

```java
public class BackgroundMusicService extends Service implements MediaPlayer.OnCompletionListener {

    MediaPlayer mediaPlayer;

    private final IBinder binder = new BackgroundMusicBinder();
    @Override
    public IBinder onBind(Intent intent) {
        return binder;
    }

    @Override
    public void onCompletion(MediaPlayer player) {
        stopSelf();
    }

    @Override
    public void onCreate(){
        super.onCreate();
        //mediaPlayer = MediaPlayer.create(this, R.raw.tt);
        mediaPlayer.setOnCompletionListener(this);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        if(!mediaPlayer.isPlaying()){
            mediaPlayer.start();
        }
        return START_STICKY;
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        if(mediaPlayer.isPlaying()){
            mediaPlayer.stop();
        }
        mediaPlayer.release();
    }

    public class BackgroundMusicBinder extends Binder{

        public BackgroundMusicService getService(){
            return BackgroundMusicService.this;
        }
    }
```

图 3-2 背景音乐播放服务

```
private BackgroundMusicService backgroundMusicService;

private ServiceConnection conn = new ServiceConnection() {

    @Override
    public void onServiceDisconnected(ComponentName name) {
        backgroundMusicService = null;
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder binder) {
        //这里我们实例化audioService,通过binder来实现
        backgroundMusicService =
                ((BackgroundMusicService.BackgroundMusicBinder)binder).getService();

    }
};

private enum MusicPlayingCommand {
    PLAY,
    STOP
}

private void processMusicPlayingCommand(MusicPlayingCommand command){
    Intent intent = new Intent();
    intent.setClass(this, BackgroundMusicService.class);
    if(command == MusicPlayingCommand.PLAY){
        bindService(intent, conn, Context.BIND_AUTO_CREATE);
    }else if(command == MusicPlayingCommand.STOP){
        unbindService(conn);
    }
}
```

图 3-3 MainActivity.java;中启动和终止服务的代码

3.2 读取外存储卡权限的申请

在 Android 6.0 及以上版本系统中，除了在 AndroidManifest.xml 中声明权限以外，还需要在 Activity 中动态申请权限，才能保证 app 的正常运行而不会崩溃。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ink.envoy.mydiarybook">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

图 3-4 在 AndroidManifest.xml 中声明读存储卡权限

```
    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static String[] PERMISSIONS_STORAGE = {
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE };
    /**
     * Checks if the app has permission to write to device storage
     * If the app does not has permission then the user will be prompted to
     * grant permissions
     * @param activity
     */
    public static void verifyStoragePermissions(Activity activity) {
        // Check if we have write permission
        int permission = ActivityCompat.checkSelfPermission(activity,
                Manifest.permission.WRITE_EXTERNAL_STORAGE);

        if (permission != PackageManager.PERMISSION_GRANTED) {
            // We don't have permission so prompt the user
            ActivityCompat.requestPermissions(activity, PERMISSIONS_STORAGE,
                    REQUEST_EXTERNAL_STORAGE);
        }
    }
```

图 3-5 在 MainActivity 中动态申请读存储卡权限

3.3 判断 SD 卡是否存在（重要）

在申请了读存储卡权限之後，还需要判断 SD 卡是否存在，以免出现访问不存在的路径的情况，导致系统报错。需要分为两步，首先判断用户的设备是否支持存储卡，如果支持再判断是否插入了卡。下图的 isSDCardAvailable()实现了此功能。

```
private boolean isSDCardAvailable() {
    boolean isSDPresent = android.os.Environment.getExternalStorageState()
            .equals(android.os.Environment.MEDIA_MOUNTED);
    boolean isSDSupportedDevice = Environment.isExternalStorageRemovable();

    return isSDSupportedDevice && isSDPresent;
}
```

图 3-6 判断用户设备上是否存在 SD 卡

3.4 使用 ContentResolver 直接获取设备上的所有音乐文件信息

Google 官方规范推荐使用 ContentResolver 来获取系统上的文件信息，这种方式安全可靠，且容易编写代码。需要注意的是，获取完毕以后需要关闭 ContentResolver 返回的 cursor，否则会造成资源泄露。

```
@Override
protected List<MusicInfo> doInBackground(Object... objects) {
    //利用ContentResolver的query函数来查询数据，然后将得到的结果放到MusicInfo对象中，最后放到数组中
    Cursor cursor = contentResolver.query(contentUri, projection, where, null, sortOrder);
    if(cursor == null){
        throw new RuntimeException("Music Loader cursor == null");
    }else if(!cursor.moveToFirst()){
        throw new RuntimeException("Music Loader cursor.moveToFirst() returns false");
    }else{
        int displayNameCol = cursor.getColumnIndex(Media.DISPLAY_NAME);
        int albumCol = cursor.getColumnIndex(Media.ALBUM);
        int idCol = cursor.getColumnIndex(Media._ID);
        int durationCol = cursor.getColumnIndex(Media.DURATION);
        int sizeCol = cursor.getColumnIndex(Media.SIZE);
        int artistCol = cursor.getColumnIndex(Media.ARTIST);
        int urlCol = cursor.getColumnIndex(Media.DATA);
        do{
            String title = cursor.getString(displayNameCol);
            String album = cursor.getString(albumCol);
            long id = cursor.getLong(idCol);
            int duration = cursor.getInt(durationCol);
            long size = cursor.getLong(sizeCol);
            String artist = cursor.getString(artistCol);
            String url = cursor.getString(urlCol);

            MusicInfo musicInfo = new MusicInfo(id, title);
            musicInfo.setAlbum(album);
            musicInfo.setDuration(duration);
            musicInfo.setSize(size);
            musicInfo.setArtist(artist);
            musicInfo.setUrl(url);
            musicList.add(musicInfo);

        }while(cursor.moveToNext());
    }
    cursor.close();
    return musicList;
}
```

图 3-7 使用 ContentResolver 获取用户设备上的音乐文件信息

3.5 使用多线程技术 AsyncTask 来异步加载音乐信息

由于使用 ContentResolver 获得的音乐信息列表可能很大，在 UI 线程上逐一读取时会导致用户界面卡顿，严重时将导致 app 无响应被系统杀死。因此需要通过 AsyncTask 来使用多线程以解决此问题。

```
public class MusicLoader
        extends AsyncTask<Object, Integer, List<MusicLoader.MusicInfo>> {

    private static List<MusicInfo> musicList = new ArrayList<MusicInfo>();

    private static MusicLoader musicLoader;

    private static ContentResolver contentResolver;
    //Uri，指向external的database
    private Uri contentUri = Media.EXTERNAL_CONTENT_URI;
    //projection：选择的列；where：过滤条件；sortOrder：排序。
    private String[] projection = {
            Media._ID,
            Media.DISPLAY_NAME,
            Media.DATA,
            Media.ALBUM,
            Media.ARTIST,
            Media.DURATION,
            Media.SIZE
    };
    private String where
            = "mime_type in ('audio/mpeg','audio/x-ms-wma') and bucket_display_name <>
    private String sortOrder = Media.DATA;

    private static OnLoadFinishListener onLoadFinishListener;

    public interface OnLoadFinishListener {
        void onFinish(List<MusicInfo> result);
    }

    public static MusicLoader instance(ContentResolver pContentResolver) {...}

    public static MusicLoader instance(ContentResolver pContentResolver,
                                       OnLoadFinishListener listener){
        if(musicLoader == null){
            contentResolver = pContentResolver;
            musicLoader = new MusicLoader();
        }
        if(onLoadFinishListener == null) {
            onLoadFinishListener = listener;
        }
        return musicLoader;
    }
```

图 3-8 使用 AsyncTask 构建的音乐加载工具类 MusicLoader

```java
@Override
protected List<MusicInfo> doInBackground(Object... objects) {...}

@Override
protected void onPostExecute(List<MusicInfo> result) {
    onLoadFinishListener.onFinish(result);
}
```

图 3-9 异步加载相关的类成员方法（doInBackground 已在图 3-7 中给出）

3.6 在调试时，使用 adb 命令将音乐上传到 android 模拟器

连接 android 手机后，打开命令行，使用 adb devices 检查设备是否正常连接。

```
F:\androidwork\android\workplace\Advert>adb devices
List of devices attached
GB8ZAH06J0      device
192.168.19.31:5555      device


F:\androidwork\android\workplace\Advert>
```

图 3-10 检查设备正常连接的命令

检测连接正常后，使用以下命令将本地文件传输到 android 手机：

adb -s [SOCKET IP 地址] push [本地文件路径] [目标设备路径]

```
F:\androidwork\android\workplace\Advert>adb -s 192.168.19.31:5555 push C:/advert /sdcard/
C:/advert\: 1 file pushed. 0.0 MB/s (15 bytes in 0.191s)
```

图 3-11 传输文件的命令

四、软件操作流程

软件启动后，用户可以看到在 app 内曾经写过的日记，长按某条日记的卡片可以将其删除。点击右下角的圆形“+”按钮，或者右上角菜单可以创建新日记。也可以在右上角菜单清空所有日记。在书写日记时，日记的内容将实时保存，无需担心数据丢失问题。
点击右上角菜单，可以播放或停止背景音乐。

五、难点和解决方案

本次 app 设计中遇到如下问题并成功解决：

1. bindService 比 startService 的用法更复杂，使用时容易出错，需仔细编码
2. 获取读 SD 卡权限后还需要检查 SD 卡是否存在
3. 读取设备上的音乐文件信息需要使用多线程，而不能在 UI 线程中执行，否则会造成用户界面卡顿
4. 手动扫描 SD 卡上的音乐文件编码较复杂且性能低下，应使用性能更高的 ContentResolver
5. 将音乐文件上传到模拟器需要用到 adb 调试命令

六、不足之处和今后的设想

本次项目历时较短，未能投入足够多的时间设计和开发 app，虽然界面相对美观，但功能较简陋，只在上次的日记本 app 上加入了背景音乐播放功能，且不符合 I18N 的规范。
将来可在此 app 的基础上进一步增加功能，开发有实用价值的产品。

附录

参考文献

[1] 《Android 移动开发基础案例教程》，黑马程序员编著，2017，人民邮电出版社
[2] Check whether the SD card is available or not programmatically
https://stackoverflow.com/questions/7429228/check-whether-the-sd-card-is-available-or-not-programmatically
[3] Android 中播放音乐的几种方式
http://blog.csdn.net/u013366008/article/details/76577372
[4] Android 多媒体学习六：利用 Service 实现背景音乐的播放
http://blog.csdn.net/chenjie19891104/article/details/6330720
[5] ANDROID STUDIO 使用 ADB 命令传递文件到 ANDROID 设备
http://www.cnblogs.com/cq-jiang/p/7661302.html
[6] Android6.0 读写 SD 卡权限动态申请
http://blog.csdn.net/gf771115/article/details/53996989
[7] Service vs Thread in Android
https://stackoverflow.com/questions/22933762/service-vs-thread-in-android
[8] Android 中利用 ContentResolver 获取本地音乐和相片
http://blog.csdn.net/zhang31jian/article/details/21231467
[9] 详解 Android 中 AsyncTask 的使用
http://blog.csdn.net/liuhe688/article/details/6532519

Ziyan Wang
School of Computer
South China Normal University
ziyan.wang@m.scnu.edu.cn