中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师： 郑贵峰

年级	15 级	专业（方向）	软件工程（移动信息工程） 互联网方向
学号	15352211	姓名	林苗
电话	13763360840	Email	554562948@qq.com
开始日期	2017.11.30	完成日期	2017.12.1

一、 实验题目

服务与多线程——简单音乐播放器

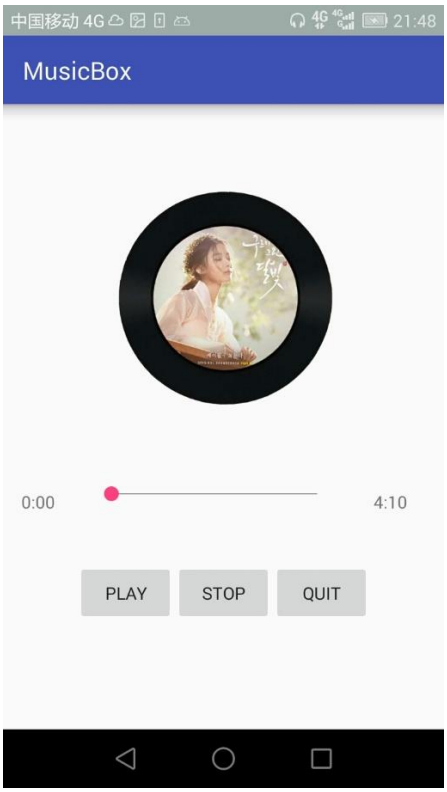
二、 实现内容

实现一个简单的播放器，要求功能有：

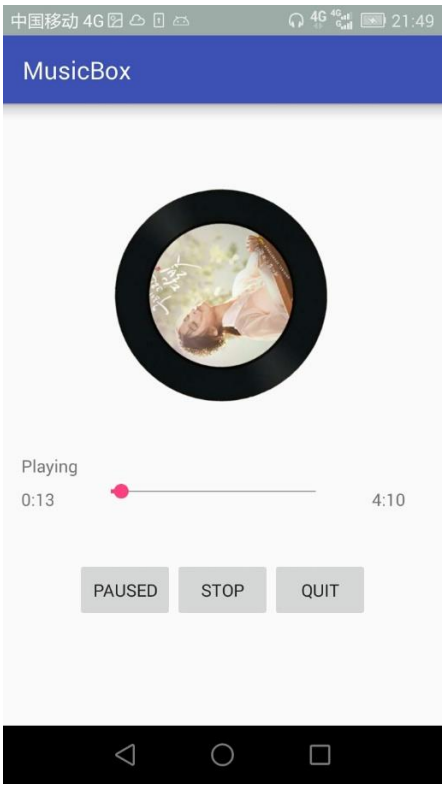
1. 播放、暂停，停止，退出功能；
2. 后台播放功能；
3. 进度条显示播放进度、拖动进度条改变进度功能；
4. 播放时图片旋转，显示当前播放时间功能；

三、 课堂实验结果

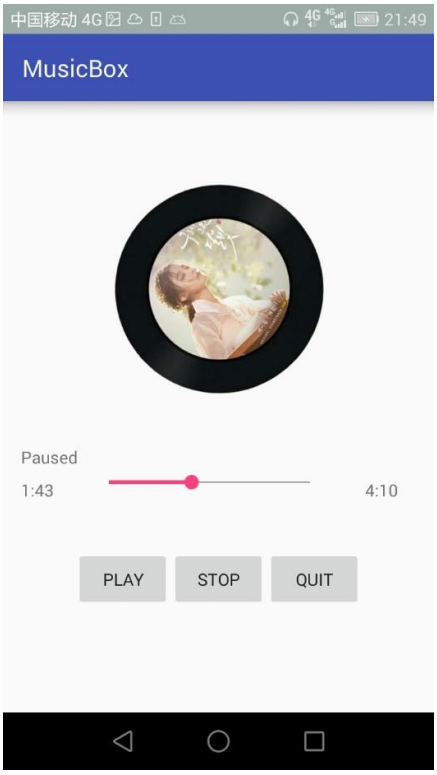
（1） 实验截图



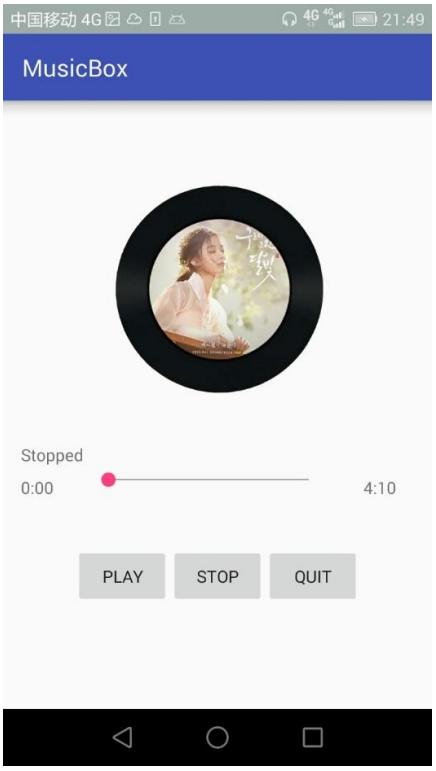
打开程序主页面



开始播放



暂停



停止

（2） 实验步骤及关键代码

1) 编写布局文件 activity_main.xml，整体采用 ConstraintLayout，滑动条和时间显示部分采用 RelativeLayout，按钮组合也采用 RelativeLayout，采用 RelativeLayout 的好处是它可以实现控件之间、控件和父元素之间的边缘对齐，满足我们对界面的 UI 需求。

2) 在 MainActivity.java 中，定义控件元素，用 findViewById 将控件元素与 xml 中的控件对应起来。

```
private Button PlayButton;
private Button StopButton;
private Button QuitButton;
private TextView CurrTime;
private TextView MusicStatus;
private TextView Length;
private ImageView Image;
private SeekBar Seekbar;
private boolean tag = false;
private int statusid = 0;
```

```
PlayButton = (Button)findViewById(R.id.play);
StopButton = (Button)findViewById(R.id.stop);
QuitButton = (Button)findViewById(R.id.quit);
CurrTime = (TextView)findViewById(R.id.currtime);
MusicStatus = (TextView)findViewById(R.id.status);
Length = (TextView)findViewById(R.id.length);
Image = (ImageView)findViewById(R.id.image);
Seekbar = (SeekBar)findViewById(R.id.seekbar);
```

3) 按键实现控制动画旋转、按键文字切换。

首先设置动画

```
/*动画旋转*/
final ObjectAnimator animator = ObjectAnimator.ofFloat(Image,"rotation",0,360);
animator.setDuration(5000); //动画时间
animator.setInterpolator(new LinearInterpolator()); //不停止
animator.setRepeatCount(-1); //不断重复
```

在 PlayButton 的监听事件中，通过 tag 来判断音乐有没有播放，通过 statusid 来判断当前是处于暂停播放还是完全没有播放的状态：

```
//没有播放
if(!tag){
    //音乐没有播放
    if(statusid==0){
        animator.start();  //动画从头开始
        statusid = 1;
    }
    //音乐暂停播放
    else{
        animator.resume();  //动画从当前停止时刻开始
    }
    MusicStatus.setText("Playing");  //播放状态
    PlayButton.setText("PAUSED");  //PLAY按钮
    tag = true;
}
//正在播放
else{
    animator.pause();  //图片旋转暂停
    MusicStatus.setText("Paused");  //播放状态
    PlayButton.setText("PLAY");  //PLAY按钮
    tag = false;
}
```

在 StopButton 的监听事件中：

```
animator.end();  //图片停止旋转
MusicStatus.setText("Stopped");  //播放状态
PlayButton.setText("PLAY");  //播放按钮
tag = false;
statusid = 0;
```

在 QuitButton 的监听事件中：

```
animator.end();  //图片停止旋转
statusid = 0;
//解除Service绑定
unbindService(MusServConn);
MusServConn = null;
stopService(stopIntent);
try{
    MainActivity.this.finish();
    System.exit(0);
}catch (Exception e){
    e.printStackTrace();
}
```

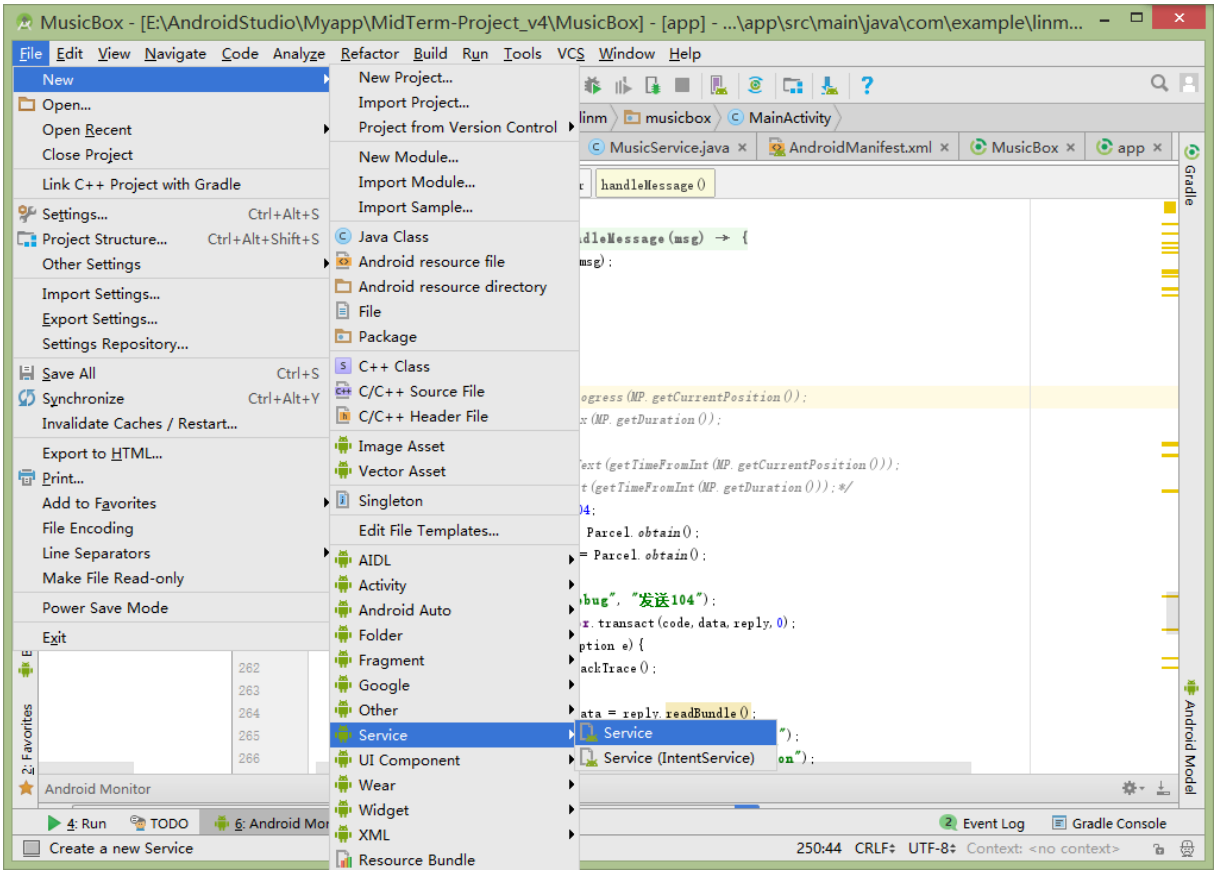
还要声明 stopIntent：

```
final Intent stopIntent = new Intent(this, MusicService.class);
```

至此就可以实现图片的旋转和控制。

4) 使用 MediaPlayer

使用如下方法创建 service：



编写 MusicService.java，在 MusicService 类中，定义：

```
private IBinder mBinder;
private MediaPlayer MP = new MediaPlayer();
```

定义构造函数，通过 setDataSource()设置音乐路径

```
public MusicService() {
    try{
        Log.e("debug", "进入初始化音乐");
        MP.setDataSource(Environment.getExternalStorageDirectory() + "/musicbox/melt.mp3");
        Log.e("debug", "成功初始化音乐");
        MP.prepare();
        MP.setLooping(true);
    }catch(Exception e){
        Log.e("debug", "初始化音乐失败");
        e.printStackTrace();
    }
}
```

编写 onCreate()函数：

```
@Override
public void onCreate() {
    super.onCreate();
    mBinder = new MyBinder();
}
```

需要注意的是，构造函数在 onCreate 之前被调用，因此构造函数中使用到的 MP 得实现定义（new），且在 onCreate 中不需要再重新 new 一遍。

重写 onBind 函数：

```
@Override
public IBinder onBind(Intent intent) { return mBinder; }
```

在 MusicService 类中，定义 MyBind 类，重写 onTransact 函数：

```
public class MyBinder extends Binder{

 @Override
   protected boolean onTransact(int code, Parcel data, Parcel reply, int flags)throws RemoteException{
```

onTransact 函数的参数中，code 表示编码，通过编码来控制 MediaPlayer 要执行何种操作；data 是 MainActivity 发过来的数据，包括拖动进度条的位置等等；reply 是 MediaPlayer 返回的数据，包括当前音乐播放的时间等等；flags 是标志位，我们此次没有使用。

onTransact 函数：

```
switch (code){
      case 101:
          //播放按钮，服务处理函数
          if (MP!=null){
              if(MP.isPlaying()){
                  MP.pause(); //暂停
              }
              else{
                  MP.start(); //开始
              }
          }
          break;
```

```
case 102:
    //停止按钮，服务处理函数
    if (MP != null) {
        MP.stop();  //停止播放
        try{
            MP.prepare();
            MP.seekTo(0);  //音乐回到初始状态
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    break;
```

```
case 103:
    //退出按钮，服务处理函数
    if (MP != null) {
        MP.reset();
        MP.release();
    }
    break;
```

在 MainActivity 中，通过 IBinder 与 service 通信：

```
//后台音乐服务
private IBinder MusBinder;
private ServiceConnection MusServConn;


//bindService成功后回调onServiceConnected函数，通过IBinder获取Service对象，实现Activity与Service的绑定
MusServConn = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        MusBinder = service;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) { MusBinder = null; }
};
//启动服务
Intent intent = new Intent(this, MusicService.class);
startService(intent);
bindService(intent,MusServConn, Context.BIND_AUTO_CREATE);
```

接下来处理各按钮与服务之间的通信。
在 PlayButton 按钮的监听事件中：

```
try{
    int code = 101;
    Parcel data = Parcel.obtain();
    Parcel reply = Parcel.obtain();
    Log.e("debug", "发送101");
    MusBinder.transact(code, data, reply, 0);

}catch (RemoteException e){
    Log.e("debug", "发送101错误");
    e.printStackTrace();
}
```

在 StopButton 按钮的监听事件中：

```
try{
    int code = 102;
    Parcel data = Parcel.obtain();
    Parcel reply = Parcel.obtain();
    Log.e("debug", "发送102");
    MusBinder.transact(code, data, reply, 0);

}catch (RemoteException e){
    Log.e("debug", "发送102错误");
    e.printStackTrace();
}
```

在 QuitButton 按钮的监听事件中：

```
try{
    int code = 103;
    Parcel data = Parcel.obtain();
    Parcel reply = Parcel.obtain();
    Log.e("debug", "发送103");
    MusBinder.transact(code, data, reply, 0);

}catch (RemoteException e){
    Log.e("debug", "发送103错误");
    e.printStackTrace();
}
```

至此我们已经能够实现按钮控制音乐的播放、暂停和停止功能了，且可以实现后台播放。点击退出按钮，程序会退出，音乐停止播放。

5) 实现滑动条更新进度、播放时间进度的更新、歌曲时长的获取。

定义一个新线程，该线程每 100 毫秒执行一次：

```
Thread mThread = new Thread() {
    @Override
    public void run() {
        while(true) {
            try{
                Thread.sleep(100);
            }catch (InterruptedException e) {
                e.printStackTrace();
            }
            if(MusServConn != null && hasPermission==true) {
                mHandler.obtainMessage(123).sendToTarget();
            }
        }
    }
};

mThread.start();
```

重写 Handler 的 handleMesseage 函数，当接收到线程发过来的“123”时，获取音乐的播放进度和时间，同时改变进度条的进度、更新时间控件的时间。

```
final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        switch (msg.what) {
            case 123:
                //UI更新内容
                //更新进度条
                /*
                Seekbar.setProgress(MP.getCurrentPosition());
                Seekbar.setMax(MP.getDuration());
                //更新时间
                CurrTime.setText(getTimeFromInt(MP.getCurrentPosition()));
                Length.setText(getTimeFromInt(MP.getDuration()));*/
```

如果不使用 IBinder 与服务进行通信的话，可以直接将服务中的音乐 MP 定义为全局静态，在 MainActivity 中直接调用（见注释部分代码），就能够获取音乐当前进度和总时长，但是我们这次要求要通过 IBinder 实现 Activity 与服务之间的通信，因此在 Activity 这边可以发送 104 编码请求服务更新时间，而 MusicService 的 IBinder 接收到编码为 104 的请求之后，通过 reply 返回所需内容，然后 Activity 这边再根据接收到的值来对控件做相应更改。

```
int code = 104;
Parcel data = Parcel.obtain();
Parcel reply = Parcel.obtain();
try{
    Log.e("debug", "发送104");
    MusBinder.transact(code, data, reply, 0);
}catch (Exception e){
    e.printStackTrace();
}
Bundle replydata = reply.readBundle();
int currpos = replydata.getInt("currpos");
int duration = replydata.getInt("duration");
//更新进度条

Seekbar.setProgress(currpos);
Seekbar.setMax(duration);
//更新时间
CurrTime.setText(getTimeFromInt(currpos));
Length.setText(getTimeFromInt(duration));

break;
```

在 MusicService 类的 IBinder 类的 onTransact 函数中，增加编码新事件实现：

```
case 104:
    //界面刷新，服务返回数据函数
    int currpos = MP.getCurrentPosition();
    int duration = MP.getDuration();
    Bundle replydata = new Bundle();
    replydata.putInt("currpos", currpos);
    replydata.putInt("duration", duration);
    reply.writeBundle(replydata);
    break;
```

这样就能实现滑动条动态更新进度和时间更新了（每 100 毫秒更新一次）。

这里需要指出的是，ppt 上说可以通过 SimpleDateFormat 来格式所需要的数据，但是我在用的时候 AS 报错说 SimpleDateFormat 只能在 API 为 24 以上的手机运行，然而我的手机的 API 是 19，因此只能用另一种方法：得知 getCurrentPosition 和 getDuration 返回的都是毫秒，因此可以手动将毫秒转换成字符串显示分钟和秒数：
在 MainActivity 中声明以下函数：

```
//时间格式
public static String getTimeFromInt(int time) {

    if (time <= 0) {
        return "0:00";
    }
    int secondnd = (time / 1000) / 60;
    int million = (time / 1000) % 60;
    String f = String.valueOf(secondnd);
    String m = million >= 10 ? String.valueOf(million) : "0"
            + String.valueOf(million);
    return f + ":" + m;
}
```

通过直接调用 getTimeFromInt 函数就可以获取带理想格式的时间字符串了。

6) 实现滑动条改变音乐进度

通过 data.writeInt 写入滑动条的进度，然后将该进度发送给服务：

```
Seekbar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        int code = 105;
        Parcel data = Parcel.obtain();
        data.writeInt(seekBar.getProgress());
        Parcel reply = Parcel.obtain();
        try{
            MusBinder.transact(code, data, reply, 0);
        }catch (RemoteException e){
            e.printStackTrace();
        }

    }
});
```

在 MusicService 类的 IBinder 类的 onTransact 函数中，增加编码新事件实现：

```
case 105:
    //拖动进度条，服务处理函数
    Log.e("debug","时长"+ MP.getDuration());
    MP.seekTo(data.readInt());
    break;
```

至此已经能够实现通过改变滑动条的进度来改变音乐播放的进度了。

7) 动态获取权限

首先要在 Manifest.xml 中注册权限，注意要写在 application 外：

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

在 MainActivity 中，初始化获取权限变量为 false：

```
//获取权限
private static boolean hasPermission = false;
```

获取 SD 卡权限：

```
/*获取Sd卡权限相关*/
public void verifyStoragePermissions()
{
    try {
        if (ActivityCompat.checkSelfPermission(this, "android.permission.READ_EXTERNAL_STORAGE") != PackageManager.PERMISSION_GRANTED)
        {
            Log.e("debug", "此处应弹出提示框");
            ActivityCompat.requestPermissions(this, new String[]{"android.permission.READ_EXTERNAL_STORAGE"}, 1);

        }else
        {
            Log.e("debug", "成功获取权限");
            hasPermission = true;
        }
    }
    catch (Exception localException)
    {
        localException.printStackTrace();
    }
    Log.e("debug", "verifyStoragePermissions()执行完成");
}
```

请求权限会弹出询问框，用户选择后，系统会调用如下回调函数：

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    Log.e("debug", "系统回调，尝试获取权限");
    if(grantResults.length>0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
        //用户同意获取权限，修改hasPermission
        Log.e("debug", "系统回调成功获取权限");
        hasPermission = true;
    }else{
        //若用户拒绝获取权限
        Toast.makeText(this, "用户拒绝权限", Toast.LENGTH_SHORT).show();
        finish();
    }

}
```

在 onCreate 函数中，调用权限获取函数：

```
//获取权限
verifyStoragePermissions();
```

由于只有对于安卓 6.0 以上的机型才需要动态获取文件阅读权限，而我的手机是 4.4.2，因此并不会弹框，但是在 debug 信息中可以看到：

```
12-04 09:18:12.610 8795-8795/com.example.linm.musicbox E/debug: 成功获取权限
12-04 09:18:12.610 8795-8795/com.example.linm.musicbox E/debug: verifyStoragePermissions()执行完成
```

说明在 verifyStoragePermission 中已经成功获取权限了，因此没有弹出框，也不会调用回调函数。

8) 返回键实现 home 键功能

一开始按返回键时，后台服务仍能够正常运行，但是点开 app，发现 UI 界面又回到初始化了，做以下更改可以实现返回键等同于 home 键功能。

在 AndroidManifest.xml 中，将 MainActivity 的模式定义为 singleInstance

```
<activity android:name=".MainActivity"
    android:launchMode="singleInstance">
```

在 MainActivity.java 中，加入如下代码：

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode== KeyEvent.KEYCODE_BACK){
        moveTaskToBack(true);
        return false;
    }
    return super.onKeyDown(keyCode, event);
}
```

（3） 实验遇到的困难及解决思路

1. 图片每旋转一次之后卡顿。

解决思路：将 ofFloat 的第 5 个参数去掉

```
final ObjectAnimator animator = ObjectAnimator.ofFloat(Image,"rotation",0,360);
```

2. 暂停之后按开始，图片又会回到初始位置。

解决思路：用一个值表示处于那种状态，如果是处于 paused 状态，那么点击 play 的话使用 resume()，如果不是处于 paused 状态，点击 play 的监听事件使用 start()。

3. 音乐路径正确，但是无法播放音乐。

解决思路：报错信息是 error(-38,0)，百度了之后发现是找不到音乐源，但是确保路径是正确的。重新看了自己的代码发现 MediaPlayer MP 新建了两次

```
private  MediaPlayer MP =new MediaPlayer();

public MusicService() {
    try{
        Log.e("debug", "进入初始化音乐");
        MP.setDataSource(Environment.getExternalStorageDirectory() + "/musicbox/melt.mp3");
        Log.e("debug", "成功初始化音乐");
        MP.prepare();
        MP.setLooping(true);
    }catch(Exception e){
        Log.e("debug", "初始化音乐失败");
        e.printStackTrace();
    }
}
@Override
public void onCreate(){

    super.onCreate();
    Log.e("debug", "创建musicplayer");
    MP = new MediaPlayer();
    mBinder = new MyBinder();
}
```

遂将第一个 new 去掉，发现 debug 信息是初始化音乐失败。调用构造函数 MusicService()要比 onCreate()早，此时 MP 没有初始化，因此无法设置源。因此将 onCreate()中的 MP = new MediaPlayer();注释掉，发现就可以成功播放音乐了！

4. SimpleDateFormat 用于显示时间，但是只能用于 API24 以上，而我的手机的 API 是 19。

解决思路：通过函数 getDuration()：获取流媒体的总播放时长，单位是毫秒。
getCurrentPosition()：获取当前流媒体的播放的位置，单位是毫秒。
可以用函数自行转换成分秒。

四、 实验思考及感想

在这次简单音乐播放器的实验中，主要学习接触了动画、MediaPlayer 的使用、服务、多线程、滑动条 SeekBar 的使用。下面简单说一下服务和多线程的大致流程。

首先要定义一个新的 Service 类,Service 类中注册 MediaPlayer 和自定义 MyBinder,在 MyBinder 类中重写 onTransact 函数，定义接收不同编码时对 MediaPlayer 的操作。

在 MainActivity 中，要定义 IBinder 类对象 MusBinder 和 ServiceConnection 类 MusServConn，然后启动服务和绑定服务，bindService 成功后回调，指定 MusBinder 获取 service 对象，在相应位置通过调用 MusBinder 的 onTransact 函数可以实现不同服务功能。

显然，onCreate 函数主要实现的是按钮的事件监听，因此对滑动条、时间更新需要在另一个线程中实现。Handler 与 UI 是同一个线程，可以通过 Handler 来更新 UI 上组件的状态。

多线程则是在 onCreate 函数中定义一个 Thread 类新线程，重写 run 函数，指定周期性调用 Handler，Handler 通过接收到的编码对 UI 做相应操作。

实现了一个简单的音乐播放器，听到音乐能如期播放时，内心还是有小小的满足的，在这个基础上也能够再实现一些拓展功能。在实验的过程中发现还是有很多细节点值得去深究的，比如四种 launchmode 的区别等等，需要学习的东西还有很多，仍需继续努力！

五、 参考资料

ObjectAnimator 动画旋转 暂停时停止在旋转到的位置
http://blog.csdn.net/qiiiiiiq/article/details/52248458

自定义控件三部曲之动画篇(七)——ObjectAnimator 基本使用
http://blog.csdn.net/harvic880925/article/details/50598322

android 开发中经常看到@Override 原来是这个意思
http://blog.csdn.net/dj0379/article/details/49779719

深入 Android MediaPlayer 的使用方法详解
http://blog.csdn.net/u011558902/article/details/41013505

关于 Parcel 的使用
http://blog.csdn.net/rainbowchou/article/details/54294394

android 时间的处理 将毫秒转化成 几分几秒
http://blog.csdn.net/liujianminghero/article/details/42042927