**湖南科技大学计算机科学与工程学院**

**智能终端系统及应用开发课程设计报告**

题目： 音乐播放器

学号： 1505040117

班级： 物联网一班

姓名： 李东

指导老师： 黄卫红

一、编写环境

*电脑版本：Win10 1703*

*编译软件：Android Studio 3.0.2*

*模拟器：自己手机Oneplus 5 安卓8.0*

*安卓API：27*

二、目的和要求

*1、掌握MedioPlayer及其使用。*

*2、熟练掌握布局及常用控件Button、ListView，SeekBar，ImageButton、ImageView、TextView等。*

*3、学着编写自定义View。*

*4、熟练掌握安卓的四大组件。*

三、功能需求

编写一个本地播放器，希望能够实现一下功能：

1. *能够通过Android自带的ContentProvide获取sdcard中所有音乐文件。并用ListView将所有音乐的音乐名在开始界面进行展示。*
2. *在主界面点击一首歌进行播放，然后在播放的activity中调用PlayService进行对播放的控制。*
3. *用Seekbar来对音乐播放的进度进行展示和控制。*
4. *用一个自定义View来在播放界面展示声音的高低音变化。*
5. *本地音乐排序（增加对中文歌的处理）*
6. *本地音乐搜索（未实现）*
7. *拨打电话或者被拨打进来电话要暂停音乐播放。*

程序运行流程图如下：

歌曲展示，并且在底部状态栏显示播放器此时状态

选择一首歌曲播放

当前是否有歌曲正在播放

是

暂停当前歌曲，播放选中歌曲，跳到播放控制界面，播放完成，自动跳到下一首

否

播放选中歌曲，跳到播放控制界面，播放完成自动跳到下一首

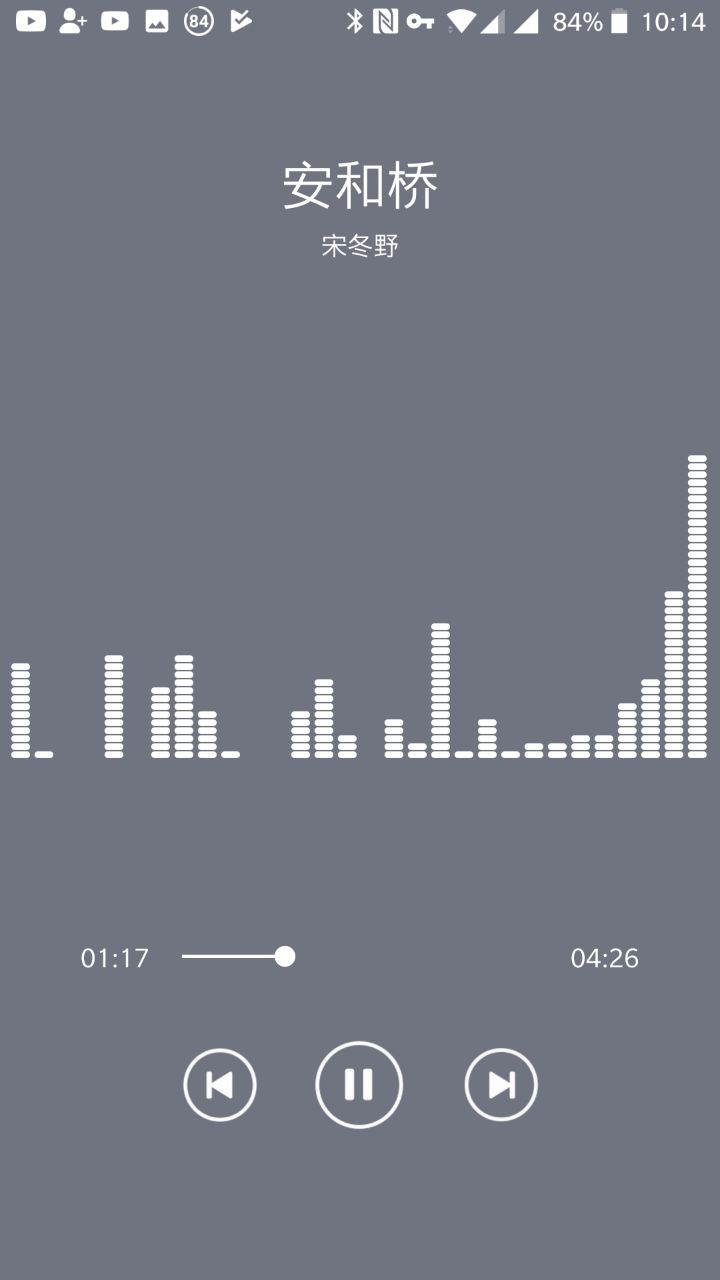
三、设计过程

I、总体说明

(1)音乐播放器寻找音乐的模块通过安卓自带的AudioManage，这个ContentProvide记录了sdcard上所有的媒体文件，我们只要用cursor遍历，然后把音乐文件信息加到MusicList这个ListView中。

(2)播放界面顶部是歌名和歌手名，下面是自定义View，然后有个SeekBar展示和控制播放进度，然后三个ImageButton,中间那个Imagebutton根据歌曲播放状态更改图片资源为暂停或者播放图片。

II、界面设计



主界面 播放界面

***III、类的核心代码***

***1、music类***

**package** **cn.lidongspace.musiclol.datautil**;

*/\**

*音乐类的定义，包括了名字，路径，和歌手是谁*

*\*/*

**public** **class** **Music** {

**private** String name;//音乐名字

其他定义省略

Music(String name, String path, String artist) {

setName(name);

setPath(path);

setArtist(artist);

}

**public** String getArtist() {

**return** artist;

}

其他获取方式省略

**public** void setArtist(String artist) {

**this**.artist = artist;

}

其他设置方式省略

}

***2、musicList类***

*/\**

*从安卓自带的mediacontendprovide服务里获取所有音乐的信息*

*\*/*

**public** **class** **MusicList** {

**private** void initMusicList() {

**if** (lists == **null**) {

lists = **new** ArrayList<>();

createMusicList();

} **else** {

**if** (lists.isEmpty()) {

createMusicList();

}

}

}

**public** ArrayList<Music> getList() {

initMusicList();

**return** **new** ArrayList<>(lists);

}

//遍历，把音乐加到ListView中

**private** void createMusicList() {

Cursor cursor = context.getContentResolver().query(

MediaStore.Audio.Media.EXTERNAL\_CONTENT\_URI, **null**, **null**, **null**, **null**);

String name, path, artist;

**if** (cursor != **null**) {

**while** (cursor.moveToNext()) {

name = cursor.getString(cursor.getColumnIndex(Media.TITLE));

artist = cursor.getString(cursor.getColumnIndex(Media.ARTIST));

path = cursor.getString(cursor.getColumnIndex(Media.DATA));

int duration = cursor.getInt(cursor.getColumnIndex(Media.DURATION));

int isMusic = cursor.getInt(cursor.getColumnIndex(Media.IS\_MUSIC));

**if** (isMusic != 0 && duration > 60000)

lists.add(**new** Music(name, path, artist));

}

}

Collections.sort(lists, cmp);

**if** (cursor != **null**) {

cursor.close();

}

}

//对比模块，对音乐进行排序

**private** Comparator<Music> cmp = **new** Comparator<Music>() {

**public** int compare(Music o1, Music o2) {

String str1 = o1.getName();

String str2 = o2.getName();

**if** (isEnglish(str1) && isEnglish(str2))

**return** str1.compareTo(str2);

**else** **if** (!isEnglish(str1) && !isEnglish(str2)) {

String[] str = {str1, str2};

Comparator<Object> com = Collator.getInstance(java.util.Locale.CHINA);

Arrays.sort(str, com);

**if** (o1.getName().equals(str[0]))

**return** -1;

**return** 1;

} **else** {

**if** (!isEnglish(str1))

str1 = getHeadChar(str1);

**if** (!isEnglish(str2))

str2 = getHeadChar(str2);

**return** str1.compareTo(str2);

}

}

};

**private** boolean isEnglish(String str) {

char[] a = str.toCharArray();

**return** (a[0] >= 'a' && a[0] <= 'z') || (a[0] >= 'A' && a[0] <= 'Z');

}

**private** String getHeadChar(String str) {

**return** **new** ChinaInitial().getPyHeadStr(str, **true**);

}

**private** **static** ArrayList<Music> lists;

}

***3、ChinaInitial类***

**package** **cn.lidongspace.musiclol.datautil**;

*/\**

*如果遇到中文歌名的排序方法*

*\*/*

**public** **class** **ChinaInitial** {

*/\*\**

*\* 返回中文拼音首字母缩写*

*\*/*

**public** String getPyHeadStr(String strChinese, boolean bUpCase) {

**try** {

StringBuffer buffer = **new** StringBuffer();

byte b[] = strChinese.getBytes("GBK");

**for** (int i = 0; i < b.length; i++) {

**if** ((b[i] & 255) > 128) {

int char1 = b[i++] & 255;

char1 <<= 8;

int chart = char1 + (b[i] & 255);

buffer.append(getIndexChar((char) chart, bUpCase));

**continue**;

}

char c = (char) b[i];

**if** (!Character.isJavaIdentifierPart(c))

c = 'A';

buffer.append(c);

}

**return** buffer.toString();

} **catch** (Exception e) {

e.getStackTrace();

}

**return** **null**;

}

*/\*\**

*\* 得到首字母*

*\*/*

**private** char getIndexChar(char strChinese, boolean bUpCase) {

int charGBK = (int) strChinese;

char result;

**if** (charGBK >= 45217 && charGBK <= 45252)

result = 'A';

**else** **if** (charGBK >= 45253 && charGBK <= 45760)

result = 'B';

**else** **if** (charGBK >= 47297 && charGBK <= 47613)

以此类推，省略

**if** (!bUpCase)

result = Character.toLowerCase(result);

**return** result;

}

}

***4、ListActivity***

**/\***

**展示歌曲名在主界面的activity**

**\*/**

**public** **class** **ListActivity** **extends** Activity **implements** View.OnClickListener {

**@Override**

**protected** void onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.activity\_list);

**if** (hasPermissions()) {

initialize();

}

}

*//查看是否有需要的权限*

**private** boolean hasPermissions() {

**if** (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

**if** (ContextCompat.checkSelfPermission(ListActivity.this,

Manifest.permission.READ\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(**this**, **new** String[]{

Manifest.permission.READ\_EXTERNAL\_STORAGE,

其他权限请求省略

}, ALL);

**return** **false**;

}

}

**return** **true**;

}

//按钮的事件

**public** void onClick(View v) {

**switch** (v.getId()) {

**case** R.id.pauseicon:

**if** (Config.musicPlaying) CtrlPlayer.getInstance(**this**).pause();

**else** CtrlPlayer.getInstance(**this**).start();

**break**;

**case** R.id.runname:

Intent playing = **new** Intent(**this**, PlayActivity.class);

startActivity(playing);

**break**;

}

}

//对按钮设置监听

**private** void setOnClickListener() {

musicName.setOnClickListener(**this**);

musicStatus.setOnClickListener(**this**);

listView.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {

**@Override**

**public** void onItemClick(AdapterView<?> parent, View view, int position, long id) {

CtrlPlayer.getInstance(ListActivity.this).choose(position);

}

});

listView.setOnItemLongClickListener(**new** AdapterView.OnItemLongClickListener() {

**@Override**

**public** boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l) {

DialogFactory.getInstance(ListActivity.this).tipsDialog(

getString(R.string.file\_path),

MusicList.getInstance(ListActivity.this).getList().get(i).getPath(),

getString(R.string.shutdown)

);

**return** **true**;

}

});

}

**@Override**

//初始化服务

**private** void initializeService() {

**if** (!Config.serviceRunning) {

Config.serviceRunning = **true**;

Intent intent = **new** Intent().setClass(**this**, PlayService.class);

**this**.startService(intent);

**if** (!MusicList.getInstance(**this**).getList().isEmpty()) {

Config.curPosition = FileCtrl.getInstance(**this**).getPosition();

String path = MusicList.getInstance(**this**).getList().get(Config.curPosition).getPath();

CtrlPlayer.getInstance(**this**).prepare(path);

CtrlPlayer.getInstance(**this**).quickRun(FileCtrl.getInstance(**this**).getProcess());

}

}

}

//刷新自定义View

**public** void reFreshView() {

boolean flag = MusicList.getInstance(**this**).getList().isEmpty();

musicName.setText(flag ? getString(R.string.none\_music) :

MusicList.getInstance(**this**).getList().get(Config.curPosition).getName());

musicStatus.setImageResource(Config.musicPlaying ? R.drawable.main\_run :

R.drawable.main\_pause);

}

**public** **static** ListActivity getListAty() {

**return** listActivity;

}

}

***5、PlayActivity***

**public** **class** **PlayActivity** **extends** Activity **implements** View.OnClickListener {

**@Override**

**public** void onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

setContentView(R.layout.activity\_play);

PlatAty = **this**;

initViewAndResource();

**try** {

initAudioFxUi();

} **catch** (Exception e) {

e.printStackTrace();

}

nextImg.setOnClickListener(**this**);

preImg.setOnClickListener(**this**);

pauseImg.setOnClickListener(**this**);

seekBar.setOnSeekBarChangeListener(**new** SeekBar.OnSeekBarChangeListener() {

**private** boolean preIsMusicPlay;

**@Override**

**public** void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

**if** (fromUser) {

CtrlPlayer.getInstance(PlatAty).quickRun(progress);

}

}

**@Override**

**public** void onStartTrackingTouch(SeekBar seekBar) {

preIsMusicPlay = musicPlaying;

freshTime = TIME\_SHORT;

**if** (preIsMusicPlay) {

CtrlPlayer.getInstance(PlatAty).pause();

}

}

**@Override**

**public** void onStopTrackingTouch(SeekBar seekBar) {

freshTime = TIME\_LONG;

**if** (preIsMusicPlay) {

CtrlPlayer.getInstance(PlatAty).start();

}

}

});

}

**@Override**

**public** void onClick(View v) {

**switch** (v.getId()) {

**case** R.id.playing\_next:

CtrlPlayer.getInstance(**this**).next();

threadSleep(TIME\_SHORT);

**break**;

**case** R.id.playing\_pause:

**if** (musicPlaying) CtrlPlayer.getInstance(**this**).pause();

**else** CtrlPlayer.getInstance(**this**).start();

threadSleep(TIME\_SHORT);

**break**;

**case** R.id.playing\_pre:

CtrlPlayer.getInstance(**this**).pre();

threadSleep(TIME\_SHORT);

**break**;

}

}

**private** void initViewAndResource() {

handler = **new** Handler() {

**@Override**

**public** void handleMessage(Message msg) {

**if** (msg.what == FRESH)

freshSeekBar();

}

};

nextImg = (ImageView) findViewById(R.id.playing\_next);

nextImg.setImageResource(R.drawable.playing\_next);

pauseImg = (ImageView) findViewById(R.id.playing\_pause);

preImg = (ImageView) findViewById(R.id.playing\_pre);

preImg.setImageResource(R.drawable.playing\_pre);

freshTextView();

startThread();

}

**public** void freshTextView() {

name.setText(MusicList.getInstance(**this**).getList().get(curPosition).getName());

seekBar.setMax(CtrlPlayer.getInstance(**this**).getMaxTime());

artist.setText(MusicList.getInstance(**this**).getList().get(curPosition).getArtist());

**if** (musicPlaying) pauseImg.setImageResource(R.drawable.playing\_run);

**else** pauseImg.setImageResource(R.drawable.playing\_pause);

}

**private** void freshSeekBar() {

seekBar.setProgress(CtrlPlayer.getInstance(**this**).getCurTime());

maxTime.setText(calcTime(CtrlPlayer.getInstance(**this**).getMaxTime()));

curTime.setText(calcTime(CtrlPlayer.getInstance(**this**).getCurTime()));

}

**private** **class** **SeekBarThread** **implements** Runnable {

**@Override**

**public** void run() {

**while** (**true**) {

**try** {

handler.sendEmptyMessage(FRESH);

Thread.sleep(freshTime);

} **catch** (InterruptedException e) {

startThread();

}

}

}

}

**private** void initAudioFxUi() {

setVolumeControlStream(AudioManager.STREAM\_MUSIC);

LinearLayout mLayout = (LinearLayout) findViewById(R.id.audio);

mLayout.setOrientation(LinearLayout.VERTICAL);

AudioView mAudio = **new** AudioView(**this**);

float VISUALIZER\_HEIGHT\_DIP = 200f;

mAudio.setLayoutParams(**new** ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT,

(int) (VISUALIZER\_HEIGHT\_DIP \* getResources().getDisplayMetrics().density)));

mLayout.addView(mAudio);

visualizer = **new** Visualizer(CtrlPlayer.getInstance(**this**).getSessionId());

visualizer.setCaptureSize(Visualizer.getCaptureSizeRange()[1]);

mAudio.setVisualizer(visualizer);

equalizer = **new** Equalizer(0, CtrlPlayer.getInstance(**this**).getSessionId());

equalizer.setEnabled(**true**);

visualizer.setEnabled(**true**);

}

***6、Config类***

**public** **class** **Config** {

**public** **static** int curPosition;

**public** **static** boolean serviceRunning;

**public** **static** boolean musicPlaying;

**public** **final** **static** int NEXT = 1;

**public** **final** **static** int PRE = -1;

**public** **final** **static** int FRESH = 0xfff;

**public** **final** **static** int TIME\_SHORT = 30;

**public** **final** **static** int TIME\_LONG = 200;

**public** **final** **static** String POSITION = "Position";

**public** **final** **static** String PROCESS = "Process";

**public** **final** **static** String FILE\_PLAYINFO = "play.play";

**public** **final** **static** int COLOR = 0xff6f7480;

}

***7、自定义MedioPlayer（CltrPlayer）***

*/\**

*自定义Player*

*\*/*

**public** **class** **CtrlPlayer** {

**public** void choose(int p) {

*//更换歌曲*

**if** (p != Config.curPosition) {

**if** (Config.musicPlaying)

pause();

Config.curPosition = p;

prepare(MusicList.getInstance(context).getList().get(p).getPath());

start();

}

}

**public** void pause() {

Config.musicPlaying = **false**;

PlayService.pause();

}

**public** void start() {

Config.musicPlaying = **true**;

PlayService.start();

}

**public** void prepare(String path) {

PlayService.prepare(path);

Config.musicPlaying = **false**;

}

**private** void change(int ride) {

boolean preIsMusicPlay = Config.musicPlaying;

**if** (Config.musicPlaying) pause();

Config.curPosition = (Config.curPosition + ride) % MusicList.getInstance(context).getList().size();*//更换播放歌曲* prepare(MusicList.getInstance(context).getList().get(Config.curPosition).getPath());

**if** (preIsMusicPlay) start();

}

}

1. 自定义View（AudioView）

*/\**

*点击一首歌播放后进入的界面时候的样式，本来想做和网易云一样用专辑图片旋转的，后来还是做成这个音阶符*

*\*/*

**public** **class** **AudioView** **extends** View **implements** Visualizer.OnDataCaptureListener {

**private** **static** **final** int DN\_W = 480;

**private** **static** **final** int DN\_H = 160;

**private** **static** **final** int DN\_SL = 8;

**private** **static** **final** int DN\_SW = 3;

**private** **static** **final** int COLOR = Color.parseColor("#ffffff");

**private** int hGap = 0;

**private** int vGap = 0;

**private** int levelStep = 0;

**private** float strokeWidth = 0;

**private** float strokeLength = 0;

**protected** **final** **static** int MAX\_LEVEL = 40;

**protected** **final** **static** int CYLINDER\_NUM = 30;

**protected** Visualizer mVisualizer = **null**;

**protected** Paint mPaint = **null**;

**protected** byte[] mData = **new** byte[CYLINDER\_NUM];

boolean mDataEn = **true**;

**public** AudioView(Context context) {

**super**(context);

mPaint = **new** Paint();

mPaint.setAntiAlias(**true**);

mPaint.setColor(COLOR);

mPaint.setStrokeJoin(Join.ROUND);

mPaint.setStrokeCap(Cap.ROUND);

}

**@Override**

**protected** void onLayout(boolean changed, int left, int top, int right, int bottom) {

**super**.onLayout(changed, left, top, right, bottom);

float w, h, xr, yr;

w = right - left;

h = bottom - top;

xr = w / (float) DN\_W;

yr = h / (float) DN\_H;

strokeWidth = DN\_SW \* yr;

strokeLength = DN\_SL \* xr;

hGap = (int) ((w - strokeLength \* CYLINDER\_NUM) / (CYLINDER\_NUM + 1));

vGap = (int) (h / (MAX\_LEVEL + 2));

mPaint.setStrokeWidth(strokeWidth);

}

**protected** void drawCylinder(Canvas canvas, float x, byte value) {

**if** (value < 0) value = 0;

**for** (int i = 0; i < value; i++) {

float y = getHeight() - i \* vGap - vGap;

canvas.drawLine(x, y, x + strokeLength, y, mPaint);

}

}

**@Override**

**public** void onDraw(Canvas canvas) {

**for** (int i = 0; i < CYLINDER\_NUM; i++) {

drawCylinder(canvas, strokeWidth / 2 + hGap + i \* (hGap + strokeLength), mData[i]);

}

}

*/\*\**

*\* 它设置视图的可视化。退出程序时，请将启动器设置为空。*

*\* 可视化器它是可视化器设置。*

*\*/*

**public** void setVisualizer(Visualizer visualizer) {

**if** (visualizer != **null**) {

**if** (!visualizer.getEnabled()) {

visualizer.setCaptureSize(Visualizer.getCaptureSizeRange()[0]);

}

levelStep = 128 / MAX\_LEVEL;

visualizer.setDataCaptureListener(**this**, Visualizer.getMaxCaptureRate() / 2, **false**, **true**);

} **else** {

**if** (mVisualizer != **null**) {

mVisualizer.setEnabled(**false**);

mVisualizer.release();

}

}

mVisualizer = visualizer;

}

**@Override**

**public** void onFftDataCapture(Visualizer visualizer, byte[] fft, int samplingRate) {

byte[] model = **new** byte[fft.length / 2 + 1];

**if** (mDataEn) {

model[0] = (byte) Math.abs(fft[1]);

int j = 1;

**for** (int i = 2; i < fft.length; ) {

model[j] = (byte) Math.hypot(fft[i], fft[i + 1]);

i += 2;

j++;

}

} **else** {

**for** (int i = 0; i < CYLINDER\_NUM; i++) {

model[i] = 0;

}

}

**for** (int i = 0; i < CYLINDER\_NUM; i++) {

**final** byte a = (byte) (Math.abs(model[CYLINDER\_NUM - i]) / levelStep);

**final** byte b = mData[i];

**if** (a > b) {

mData[i] = a;

} **else** {

**if** (b > 0) {

mData[i]--;

}

}

}

postInvalidate();

}

**@Override**

**public** void onWaveFormDataCapture(Visualizer visualizer, byte[] waveform,

int samplingRate) {

*// 不做任何动作*

}

}

1. ***自定义Dialog***

*/\**

*\*自定义dialog*

*\*/*

**public** **class** **DialogFactory** {

**public** void tipsDialog(String strTitle, String strContent, String strButton) {

**final** AlertDialog.Builder builder = **new** AlertDialog.Builder(context);

View view = View.inflate(context, R.layout.dialog\_tips, **null**);

TextView title = (TextView) view.findViewById(R.id.tips\_title);

TextView content = (TextView) view.findViewById(R.id.tips\_content);

Button cancel = (Button) view.findViewById(R.id.tips\_shutdown);

content.setText(strContent);

title.setText(strTitle);

cancel.setText(strButton);

builder.setView(view);

**final** AlertDialog dialog = builder.create();

dialog.setCancelable(**false**);

dialog.show();

cancel.setOnClickListener(**new** View.OnClickListener() {

**@Override**

**public** void onClick(View v) {

dialog.dismiss();

}

});

}

}

***10、ListView的Adapter（ListAdapter）***

**public** **class** **ListAdapter** **extends** BaseAdapter {

**@Override**

**public** View getView(int position, View convertView, ViewGroup parent) {

ViewHolder viewHolder;

**if** (convertView == **null**) {

convertView = LayoutInflater.from(context).inflate(R.layout.item\_music,**null**);

viewHolder = **new** ViewHolder();

viewHolder.musicName = (TextView) convertView.findViewById(R.id.musicname);

viewHolder.musicName.setText(lists.get(position).getName());

convertView.setTag(viewHolder);

} **else** {

viewHolder = (ViewHolder) convertView.getTag();

viewHolder.musicName.setText(lists.get(position).getName());

}

**return** convertView;

}

**private** **static** **class** **ViewHolder** {

TextView musicName;

}

}

***11、PlayService***

*/\**

*写的播放的一个service，这样写是让播放可以再后台进行，而且将对音乐播放的控制一起放到这个服务里方便控制*

*\*/*

**public** **class** **PlayService** **extends** Service {

**private** **static** PlayService playService;

**private** **static** MediaPlayer mediaPlayer = **new** MediaPlayer();

**@Override**

**public** IBinder onBind(Intent intent) {

**return** **null**;

}

**@Override**

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

playService = **this**;

mediaPlayer.setOnCompletionListener(**new** MediaPlayer.OnCompletionListener() {

**@Override**

**public** void onCompletion(MediaPlayer mp) {

CtrlPlayer.getInstance(PlayService.this).completion();

freshAllView();

}

});

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

}

**public** **static** int getAudioSessionId() {

**try** {

**return** mediaPlayer.getAudioSessionId();

} **catch** (Exception e) {

**return** 0;

}

}

**public** **static** void quickRun(int t) {

mediaPlayer.seekTo(t);

}

**public** **static** int getDuration() {

**return** mediaPlayer.getDuration();

}

**public** **static** int getCurDuration() {

**return** mediaPlayer.getCurrentPosition();

}

**public** **static** void prepare(String path) {

**try** {

mediaPlayer.reset();

mediaPlayer.setDataSource(path);

mediaPlayer.prepare();

} **catch** (Exception e) {

e.printStackTrace();

}

freshAllView();

}

**public** **static** void freshAllView() {

**try** {

FileCtrl.getInstance(playService).saveUserInfo();

ListActivity.getListAty().reFreshView();

PlayActivity.getPlatAty().freshTextView();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** void start() {

mediaPlayer.start();

freshAllView();

}

**public** **static** void pause() {

mediaPlayer.pause();

freshAllView();

}

**private** void freePlayer() {

**if** (mediaPlayer != **null**) {

mediaPlayer.stop();

mediaPlayer.release();

mediaPlayer = **null**;

}

}

**@Override**

**public** void onDestroy() {

freePlayer();

**super**.onDestroy();

}

}

***12、TelRecive***

*/\**

*写了一个当有电话打进来的时候停止播放的Receiver*

*\*/*

**public** **class** **TelReceiver** **extends** BroadcastReceiver {

**@Override**

**public** void onReceive(Context context, Intent intent) {

**this**.context = context;

preMusicPlaying = Config.musicPlaying;

**if** (preMusicPlaying) {

CtrlPlayer.getInstance(context).pause();

}

TelephonyManager tm = (TelephonyManager) context.getSystemService(Service.TELEPHONY\_SERVICE);

tm.listen(listener, PhoneStateListener.LISTEN\_CALL\_STATE);

**if** (tm.getCallState() == TelephonyManager.CALL\_STATE\_IDLE) {

**if** (preMusicPlaying) {

**if** (!Config.musicPlaying) {

**try** {

Thread.sleep(1500);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

CtrlPlayer.getInstance(context).start();

}

}

}

}

PhoneStateListener listener = **new** PhoneStateListener() {

**@Override**

**public** void onCallStateChanged(int state, String incomingNumber) {

**super**.onCallStateChanged(state, incomingNumber);

**switch** (state) {

**case** TelephonyManager.CALL\_STATE\_IDLE:

**if** (preMusicPlaying)

**if** (!Config.musicPlaying)

CtrlPlayer.getInstance(context).start();

**break**;

}

}

};

}

***13、AudioView里的一个重要组件***

**public** **class** **VirtualKey** {

*/\*\**

*\* 关联要监听的视图*

*\*/*

**public** **static** void assistActivity(View viewObserving) {

**new** VirtualKey(viewObserving);

}

**private** View mViewObserved;*//被监听的视图*

**private** int usableHeightPrevious;*//视图变化前的可用高度*

**private** ViewGroup.LayoutParams frameLayoutParams;

**private** VirtualKey(View viewObserving) {

mViewObserved = viewObserving;

*//给View添加全局的布局监听器*

mViewObserved.getViewTreeObserver().addOnGlobalLayoutListener(**new** ViewTreeObserver.OnGlobalLayoutListener() {

**public** void onGlobalLayout() {

resetLayoutByUsableHeight(computeUsableHeight());

}

});

frameLayoutParams = mViewObserved.getLayoutParams();

}

**private** void resetLayoutByUsableHeight(int usableHeightNow) {

*//比较布局变化前后的View的可用高度*

**if** (usableHeightNow != usableHeightPrevious) {

*//如果两次高度不一致*

*//将当前的View的可用高度设置成View的实际高度*

frameLayoutParams.height = usableHeightNow;

mViewObserved.requestLayout();*//请求重新布局*

usableHeightPrevious = usableHeightNow;

}

}

*/\*\**

*\* 计算视图可视高度*

*\*/*

**private** int computeUsableHeight() {

Rect r = **new** Rect();

mViewObserved.getWindowVisibleDisplayFrame(r);

**return** (r.bottom - r.top);

}

}

14、获取文件信息类

**public** **class** **FileCtrl** {

**private** Context context;

**private** String division = "\_\_\_";

**private** FileCtrl(Context context) {

**this**.context = context;

}

**public** **static** FileCtrl getInstance(Context context) {

**return** **new** FileCtrl(context);

}

**public** int getProcess() {

**return** getFileData(Config.PROCESS, Config.FILE\_PLAYINFO);

}

**public** int getPosition() {

int result = getFileData(Config.POSITION, Config.FILE\_PLAYINFO);

**if** (result >= MusicList.getInstance(context).getList().size())

result = MusicList.getInstance(context).getList().size() - 1;

**return** result;

}

**public** void saveUserInfo() {

**try** {

FileOutputStream fOut = context.openFileOutput(Config.FILE\_PLAYINFO, MODE\_PRIVATE);

fOut.write((Config.POSITION + division + Config.curPosition + "\r\n").getBytes());

fOut.write((Config.PROCESS + division + CtrlPlayer.getInstance(context).getCurTime() + "\r\n").getBytes());

fOut.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**private** int getFileData(String infoName, String fileName) {

int result = 0;

**try** {

FileInputStream fin = context.openFileInput(fileName);

InputStreamReader isr = **new** InputStreamReader(fin);

BufferedReader br = **new** BufferedReader(isr);

String line;

**while** ((line = br.readLine()) != **null**) {

String[] dataS = line.split(division);

**if** (dataS[0].equals(infoName)) {

result = Integer.valueOf(dataS[1]);

**break**;

}

}

br.close();

isr.close();

fin.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** result;

}

}

IV、未来计划

*1、实现网络下载歌词功能。*

*2、按照专辑、歌手分类的功能。*

*3、调用其它网络播放器的API，实现歌区在线播放和下载的功能。*

*4、实现多个自定义View和主题的切换。*

*5、加入喜欢歌曲列表功能*

V、设计过程

*12.12号完成大部分布局设置，并且完成了Music类，MusicList类的编写。*

*12.13~14完成ListActivity，Conifg类，ChinaInitial类和ListAdapter的编写。*

*12.15号完成自定义MedioPlayer类，自定义Dialog编写*

*12.16~17号完成PlayActivity、PlayService和TelReceiver编写*

*12.18号完成自定义View的所有功能，并且加入播放界面中*

*12.19号调试、美化整合整个程序。*

四、实验心得

*我开始选这个播放器题目的时候并没有想太多，第一天设置布局的时候设置了很多功能但是，但是发现下手的时候功能模块太多了，都不知道从哪里下手，最后放弃了一些功能，从最基本功能做起，慢慢扩展播放器的其他功能才取得了一定的进展，自定义View我是借鉴了Github的一个项目，整合进了我的项目。*

*其他的心得就是多上网查资料，扩展自己的知识面，书本上的知识实在是太浅薄了，对一些复杂的东西讲的太浅，所有都要自己一步步去摸索，还有要对安卓整个工程的架构要了解，要知道style.xml，string.xml等布局的用法。*

*对自己程序的功能逻辑要进行优化，减少或尽量减少逻辑错误。第一次编写比较大的项目一定要分好模块，每个模块功能都实现之后再整合，这样让自己思路能够更加清晰。*