Android 手机应用开发之音乐资源播放器

汪永松

摘　要：从开发者的角度对 Android 手机平台下较为流行的音乐资源播放器进行全面的功能分析，并结合 Android 平台的相关技术和应用模式，通过实际案例，详细地介绍了该类应用的开发过程。可用于指导 Android 开发者了解音乐资源播放器应用的开发模式和实现技巧。

关键词：Android 平台；手机应用；音乐资源播放器

1　应用概述

对于智能手机，总不可以缺少音乐。不仅如此，随着 3G 网络的普及和城市 Wi-Fi 网络的覆盖，手机获取音乐的途径也逐渐从本地存储卡过渡到“空中”。而这种从“空中”获取音乐资源的模式实际上是在已有业务框架（B/S）上的扩展，新增了从无线网络到互联网的连接，其业务框架如图 1 所示。

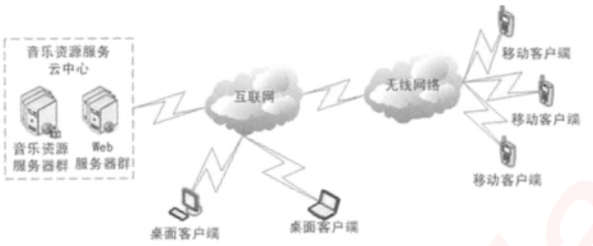

图 1　手机音乐资源播放器业务框架

在桌面应用中，其播放功能使用的是本地媒体库或包装成的播放器组件，其播放模式有两种：直接播放本地文件（下载后播放也属于此类）和以流媒体的方式播放网络资源。在第一种模式下，需要连接本地存储设备并设置媒体文件的存放路径；在第二种模式下，需要连接到互联网中音乐资源服务器，因此还需配置服务器主机和服务端口等信息。

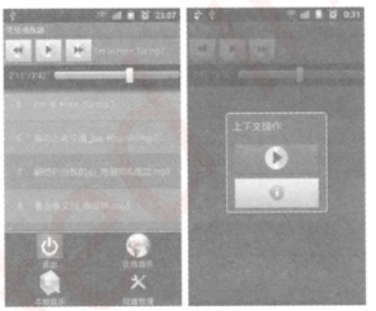

图 2　手机音乐资源播放器实机界面

对于 Android 手机应用，其播放资源也分为两种：本机存储设备上的音乐资源（主要包括：通过数据线拷贝过来或从网络下载的）和网络资源（通过 3G 网络或 Wi-Fi 网络连接到互联网进行获取）。

文中将以 Android 实机（Android 2.3.6）为验证环境，完整地介绍一款音乐资源播放器（如图 2 所示）的开发过程，希望能对 Android 开发者起到参考作用。

2　功能分析

2.1　关键功能及相关技术

通过前文对手机音乐资源播放器应用的介绍，相信读者应该可以对该类应用的功能有所了解；笔者结合自己对该类应用的研究，整理出手机音乐资源播放器的主要功能如下。

2.1.1 音乐播放控制

该功能即对音乐资源进行启动、暂停、停止、切换、定位等播放控制。其核心组件是音乐播放器，开发者一般只需调用其功能接口。Android 平台中的类“MediaPlayer”提供了媒体资源的播放控制，同时还提供了部分回调接口（侦听器）用于侦听播放状态。

作为音乐播放器，往往需要具备后台播放功能。在 Android 平台中，后台任务的执行须使用服务（Service）框架。不仅如此，用户界面（Activity 组件）还需与后台服务进行交互。在 Android 平台，Activity 组件与后台服务的交互可以采用连接或非连接的（广播）方式。

（1）连接方式，即 Activity 组件先建立与服务组件的连接（ServiceConnection），然后在该连接的回调函数中，通过调用服务接口（Interface）来实现与服务的交互。该交互方式中，Activity 是主动方，服务组件是被动方，后台服务的状态都需要 Activity 调用服务接口来获取，服务组件的状态不能主动推送给 Activity 组件。

（2）广播方式，即在 Activity 组件和服务组件中各注册广播接收器（BroadcastReceiver），然后在各组件中通过发送/接收广播的方式进行交互。该交互方式中，双方都是主动的，后台服务的状态可以主动地推送给 Activity 组件。

相比之下，第（2）种方式的交互更为灵活，本示例中采用第（2）种方式。

此外，对于手机平台，还需考虑对电话状态的监听，如果播放音乐过程中有电话呼入，则需要先暂停播放，待通话结束后再重启播放。对于电话状态的监听，Android 平台提供了电话系统管理器（TelephonyManager），使用其 API 可侦听手机的电话状态。

不仅如此，还要考虑对应用程序的唤醒（即快速呈现程序界面），该功能一般采用 Android 平台的通知（Notification）机制，可以快速地从系统状态栏调出应用程序。

2.1.2 音乐资源管理

对音乐资源的管理主要包括：获取本机和网络音乐资源信息、本地音乐资源管理、下载网络资源。本地音乐资源信息的获取和管理可以通过文件系统的 API 来实现，而网络音乐资源信息则需要通过调用服务端的信息接口，通过信息接口，移动客户端才可以获取网络音乐资源的“路径”信息（URL）。这些音乐资源信息是基础数据，本地音乐资源的管理和网络音乐资源的下载都将基于这些数据。按照 MVC 的设计原则，Android 平台提供了列表视图（ListView）和对应的适配器（Adapter）来用于数据管理和展示控制。而开发者要做的是定义音乐资源数据项和对应的适配器。

在本开发案例中，服务端采用 J2EE 平台（应用程序服务器为 Tomcat），手机端通过 Wi-Fi 网络连接到服务端。服务端提供的数据接口将以 JSON 数组格式返回服务端的音乐资源信息。

2.1.3 网络音乐资源下载

网络音乐资源的下载方式主要取决于音乐资源的提供方式，在图 1 所示的业务框架中，Web 服务器会将服务端音乐资源的 URL 提供给手机移动端，手机端通过 URL 来下载对应资源。Android 平台提供了基于 URL 的 HTTP 连接接口（HttpURLConnection），可通过该接口的 I/O 流来下载服务端资源。

对于 Android 手机平台，下载操作需要在子线程中执行（如果阻塞主线程会造成 ANR 错误），不仅如此，为了获取下载状态，需要使用消息队列、广播或通知（Notification）的机制，将线程执行信息推送给用户界面（Activity 或系统状态栏）。本示例中采用通知机制。

2.1.4 系统设置维护

系统设置维护主要用于维护设置信息，设置信息包括：服务器的主机和端口、下载音乐的存放位置、播放模式等。Android 平台提供了首选项（Preference）框架，开发者可以快捷地使用 XML 文件来定义首选项编辑界面。

2.1.5 播放列表管理

播放列表实际上是选择性地将音乐资源（无论本地资源还是网络资源）导出成列表文件，便于下次加载播放。播放列表的管理主要包括：播放列表文件的生成（包括追加）和加载。播放列表的展示也需要使用列表视图和对应的适配器。

2.2 功能结构

图 3 是该手机移动应用的功能结构图，其一级功能包括：音乐播放控制、音乐资源管理（融合播放列表管理）和设置编辑。其中音乐资源包括网络资源和本地资源，对于网络资源可以下载，对本地资源可以删除，网络资源下载完成后即作为本地资源进行管理。

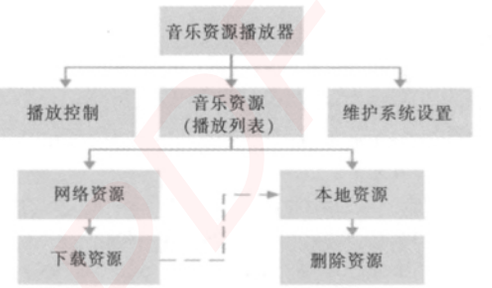

图 3 手机音乐资源播放器功能结构

3 界面设计及接口定义

3.1 界面

3.1.1 主界面

应用程序的主界面如图 2 所示，其界面内容主要包含 4 部分：播放控制面板、播放列表、选项菜单和上下文对话框。主界面的布局定义（XML）如代码 1 所示：

代码 1 音乐资源播放器主界面布局定义

文件名：layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
        android:layout_width = "fill_parent" android:layout_height="fill_parent">
    <LinearLayout android:layout_width="fill_parent"
        android:layout_height = "wrap_content" android:layout_gravity="center_vertical"
      android:background="@drawable/orange"
      android:gravity="center_vertical" android:orientation="horizontal" >
        <ImageButton android:id="@+id/btn_prev"
          android:layout_width = "wrap_content" android:layout_height="wrap_content"
            android:layout_marginTop = "2sp" android:src = "@drawable/prev" />
        <ImageButton android:id="@+id/btn_play"
          android:layout_width = "wrap_content" android:layout_height="wrap_content"
  android:layout_marginTop = "2sp" android:src = "@drawable/play" />
        <ImageButton android:id="@+id/btn_next"
```

```
                android:layout_width = "wrap_content"  android:
layout_height="wrap_content"
                android:layout_marginTop = "2sp"  android:src = "
@drawable/next" />
            <TextView android:id="@+id/text"
                android:layout_height = "wrap_content"  android:
layout_width="wrap_content"
                android:layout_gravity = "center_vertical"  android:
paddingLeft="2pt"/>
        </LinearLayout>
        <TableLayout  android:stretchColumns  =  "1"  android:
layout_gravity="center_vertical"
                 android:layout_width = "match_parent"  android:
layout_height="40sp">
        <TableRow android:layout_gravity="center_vertical"
                android:layout_marginTop="4sp">
 <TextView android:id="@+id/progress" android:text="00:00"
                        android:layout_height  =  "match_parent"
android:layout_width="90sp"
android:gravity="center_vertical" android:paddingLeft="10sp"/>
                    <SeekBar android:id="@+id/seekbar"
                android:layout_height = "match_parent"  android:
layout_width="match_parent"
                         android:gravity="center_vertical"
   android:paddingLeft="10pt" android:paddingRight="10pt"/>
            </TableRow>
        </TableLayout>
        <ListView android:id="@id/android:list"
                    android:layout_width = "fill_parent"  android:
layout_height="fill_parent"
android:fastScrollEnabled="true"  android:focusable="true"/>
            <TextView  android:id = "@id/android:empty"  android:
text="没有音乐媒体资源"
   android:layout_width = "fill_parent"  android:layout_height = "
fill_parent"/>
</LinearLayout>
```

主界面的选项菜单定义如代码 2 所示：

代码 2　音乐资源播放器主界面选项菜单定义

文件名：menu/ops_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu  xmlns:android = "http://schemas.android.com/apk/res/
android">
<item android:id="@+id/mi_quit" android:icon="@drawable/exit"
        android:title="退出" />
     <item  android:id  =  "@ +id/mi_internet"  android:icon  =  "
@drawable/internet"
        android:title="在线音乐" />
     <item  android:id  =  "@ +id/mi_local"  android:icon  =  "
@drawable/local"
        android:title="本地音乐" />
     <item  android:id  =  "@ +id/mi_setting"  android:icon  =  "
@drawable/setting"
        android:title="设置管理" />
</menu>
```

主界面的上下文对话框的布局定义如代码 3 所示：

代码 3　音乐资源播放器主界面上下文对话框布局定义

文件名：layout/ctx_dlg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
     android:layout_width = "180sp"  android:layout_height = "
wrap_content"
    android:orientation="vertical" >
    <ImageButton android:id="@+id/btn_play"
        android:layout_width = "160sp"  android:layout_height = "
wrap_content"
        android:layout_gravity="center_horizontal" android:src="
@drawable/ctx_play" />
    <ImageButton android:id="@+id/btn_info"
        android:layout_width = "160sp"  android:layout_height = "
wrap_content"
        android:layout_gravity="center_horizontal" android:src="
@drawable/ctx_info" />
</LinearLayout>
```

3.1.2 网络资源界面

网络资源界面如图 4 所示，其界面主要内容包含 3 部分：资源信息列表、选项菜单和上下文对话框。其资源列表界面的布局定义如代码 4 所示。

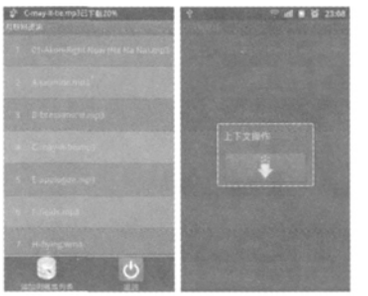

图 4　网络资源界面

代码 4　网络资源界面布局定义

文件名：layout/internet.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
    android:orientation="vertical"
    android:layout_width = "fill_parent"  android:layout_height = "
fill_parent">
        <ListView android:id="@id/android:list"
                    android:layout_width = "fill_parent"  android:
layout_height="fill_parent"
            android:fastScrollEnabled="true"/>
        <TextView android:id="@id/android:empty"
android:layout_width  =  "fill_parent"  android:layout_height  =  "
```

```
fill_parent"
            android:text="没有音乐资源" />
</LinearLayout>
```

3.1.3 本地资源界面

网络资源界面主要内容也包含 3 部分：资源信息列表、选项菜单和上下文对话框。其界面的布局定义与网络资源界面相同。

3.2 数据接口定义

本文的数据接口是指手机移动客户端获取服务端网络音乐资源信息的接口，其采用 HTTP 协议，JSON 格式（JSON 数组），数组的记录数据项定义为：

```
{title:<标题(字符串)>,uri:<资源 URI(字符串)>,size:<资源大小(整形)>}
```

例如：{"uri":"http://192.168.1.100:8080/ResServices/res/R-fields.mp3",
"title":"R-fields.mp3","size":3530126}。

4 程序框架及功能实现

4.1 程序框架

为了更好地阐明该程序的组成结构，将程序所包含的组件按照功能特性归集为 4 个部分：服务端、支撑层、数据层和展示层，后 3 层均属于移动客户端。其中，服务端主要是提供网络音乐资源信息接口；支撑层包含：播放服务、滴答服务、文件下载线程和音乐资源信息获取工具类，支撑层的音乐资源信息获取工具类调用服务端的接口来获取音乐资源信息；数据层包括音乐资源列表、系统状态栏（通知队列）和列表适配器（包装了音乐资源列表）；展示层主要是 Activity 组件，其中主 Activity 负责调用网络资源 Activity、本地资源 Activity 和设置管理 Activity，设置管理 Activity 负责调用首选项设置 Activity 和文件浏览 Activity。

图 5 程序组件关联关系

虽然各个组件被划分到不同的层，然而展示层、数据层和支撑层中的各个组件彼此都存在紧密的关联。展示层中的主 Activity、网络资源 Activity 和本地资源 Activity 中的列表视图都需要绑定音乐资源列表适配器；文件下载线程通过发送系统通知（将在系统状态栏显示）来更新下载进度；主 Activity 和播放服务通过发送广播和注册广播接收器来进行交互，主 Activity 向播放服务发送播放控制命令，播放服务向主 Activity 发送播放状态信息（当前播放音乐的信息及进度）；播放服务器负责启动滴答服务，滴答服务用于“提醒”播放服务器更新播放状态信息（发送给主 Activity）。各层组件之间的关联关系如图 5 所示。

4.2 功能实现

4.2.1 音乐播放控制

音乐播放控制由主 Activity 和播放服务共同完成。主 Activity 负责与用户进行交互，通过系统广播向播放服务发送播放控制命令；播放服务接收到命令后执行相应的播放行为。代码 5 是播放服务的创建回调函数：

代码 5 播放服务创建回调函数
文件名：PlayService.java

```
@Override
public void onCreate() {
    super.onCreate();
    //获取播放列表内容
    mPlaylist = new ArrayList<MusicRes>();
    PlaylistMgr.getInstance().getPlaylist(mPlaylist);
    if(! mPlaylist.isEmpty()) {
        mCurPos = 0;
    }
    //初始化播放器
    mPlayer = new MediaPlayer();
    mPlayer.setOnCompletionListener(this);
    mPlayer.setOnErrorListener(this);
    //启动滴答服务
    FooMonUtil.getInstance().startTickService(this);
    //注册播放控制接收器
    registerReceiver(mReceiver, mFilter);
    //初始化电话服务
    final String sname = Context.TELEPHONY_SERVICE;
mTelMgr = (TelephonyManager)getSystemService(sname);
    //设置电话状态侦听
    mListener = new FooListener();
        mTelMgr.listen (mListener, PhoneStateListener.
LISTEN_CALL_STATE);
}
```

代码 5 中，播放服务首先初始化播放器（创建播放器对象和绑定侦听器），然后启动滴答服务（用于触发更新播放状态信息）、注册播放控制命令接收器、初始化电话服务并侦听电话状态（如果有来电，将自动暂停音乐播放）。代码 6 是播放控制命令接收器的定义：

代码 6 播放控制命令接收器
文件名：PlayService.java

```
class PlayControlReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context ctx, Intent intent) {
      if(intent.getAction().compareTo(IDef.PLAY_ACTION)! =
```

```
0) { return; }
            //获取命令
            Bundle bundle = intent.getExtras();
                String command = bundle.getString (IDef.
EXTRA_COMMAND);
            int pos = bundle.getInt(IDef.EXTRA_POS);
 int progress = bundle.getInt(IDef.EXTRA_PROGRESS);
      if (command.compareToIgnoreCase (IDef.CMD_Start) ==
0) { //启动播放
                PlayService.this.mCurPos = pos;
                PlayService.this.mProgress = progress;
                //设置激活状态
                mIsActived = true;
                playDefault();
                //更新状态栏
                updateNotificationBar(); return;
            } else if (command.compareToIgnoreCase (IDef.
CMD_Stop)==0) { //停止
                PlayService.this.stopSelf(); return;
            } else if (command.compareToIgnoreCase (IDef.
CMD_Pause)==0) { //暂停
                PlayService.this.mCurPos = pos;
                PlayService.this.mProgress = progress;
                //暂停
                pausePlay(); return;
            } else if (command.compareToIgnoreCase (IDef.
CMD_Next)==0) { //后一首
                PlayService.this.mCurPos = pos;
                //DeamonService.this.mProgress = progress;
                //设置激活状态
                mIsActived = true;
                PlayService.this.playNext();
                //更新状态栏
                updateNotificationBar(); return;
            } else if (command.compareToIgnoreCase (IDef.
CMD_Prev)==0) { //前一首
                PlayService.this.mCurPos = pos;
                //DeamonService.this.mProgress = progress;
                //设置激活状态
                mIsActived = true;
                PlayService.this.playPrev();
                //更新状态栏
                updateNotificationBar(); return;
            } else if (command.compareTo(IDef.CMD_Update)
==0) { //播放列表有更新
      PlaylistMgr.getInstance ().getPlaylist (PlayService.this.
mPlaylist); return;
            } else if(command.compareTo(IDef.CMD_TICK)==
0) { //滴答命令(轮询运行状态)
                updatePlayProgress(); return;//更新播放进度
            }
    }
};
```

代码 6 中，接收器接收的命令包括：启动播放、停止、暂停、后一首、前一首、播放列表有更新和滴答命令。其中停止是指停止服务，主 Activity 也将结束；播放列表有更新是指用户更新播放列表（创建或新增播放列表）后，同步更新播放器的播放任务列表。

代码 7 是播放播放列表中指定位置的音乐资源的代码，其是播放控制的基本方法：

代码 7　播放播放列表中指定位置的音乐资源

文件名：PlayService.java

```
private boolean playMusic(int pos) {
    //判断播放序号是否有效
    if(pos == IDef.INVALID_POS) { return (false); }
    try {
        mPlayer.reset();
        mPlayer.setDataSource(mPlaylist.get(pos).getUri());
        mPlayer.prepare();
        mPlayer.start();
        mStatus = IDef.STATUS_PLAYING;
        updatePlayState();
        if(mProgress ! = IDef.INVALID_POS) {
            mPlayer.seekTo(this.mProgress);
        }
    } catch(IOException e) {
        return(false);
    }
    return(true);
}
```

代码 7 中，该方法的参数是播放列表中的序号，由此可知，播放控制中的曲目切换，实际上只需获取目标曲目在列表中的序号，再调用该方法即可。

4.2.2 音乐播放状态更新

音乐播放状态更新包括两种情形：播放曲目发生切换和播放进度发生改变。当播放曲面发生切换时，需要在系统状态栏进行提示，代码 8 是更新曲目切换状态的主要代码：

代码 8　更新曲目切换状态

文件名：PlayService.java

```
private void updateNotificationBar() {
    FooMonUtil.getInstance ().clearNotification(PlayService.
this, IDef.NOTIFICATON_ID);
    //设置通知跳转意向
    Intent i = new Intent();
    i.setComponent(new ComponentName(
        MainAct.class.getPackage().getName(), MainAct.
class.getName()));
    i.putExtra(IDef.EXTRA_POS, this.mCurPos);
        i.putExtra (IDef.EXTRA_PROGRESS, this.mPlayer.
getCurrentPosition());
      FooMonUtil.getInstance().postNotification(PlayService.
```

```
this,
                R.drawable.ic_launcher, i,
                    this.mPlaylist.get(this.mCurPos).getTitle(),
IDef.NOTIFICATON_ID);
        Log.d(IDef.APP_TAG, "更新状态栏:"+this.mCurPos);
}
```

代码 8 中，采用更新通知的方式（指定了通知 ID），将系统状态栏中通知文字改变为切换后的曲目标题，而且通过设置通知的跳转意向可实现从系统状态栏通知“唤醒”主 Activity。而当播放进度发生改变时，则需要“通知”主 Activity 更新播放进度信息，代码 9 是更新播放进度的主要代码：

代码 9　更新播放进度

文件名：PlayService.java

```
private void  updatePlayProgress() {
        if(mStatus ! = IDef.STATUS_PLAYING) { return; }
        if(mPlayer==null) { return; }
        int pos = -1;
        int duration = 0;
        try { //获取播放进度信息
                pos = mPlayer.getCurrentPosition();
                duration = mPlayer.getDuration();
        } catch(IllegalStateException e) {
                e.printStackTrace();
                return;
        }
        //设置播放进度信息并发送广播
        Intent intent = new Intent();
        intent.setAction(IDef.FEEDBACK_ACTION);
        intent.putExtra(IDef.EXTRA_STATE, mStatus);
        intent.putExtra(IDef.EXTRA_PROGRESS, pos);
        intent.putExtra(IDef.EXTRA_DURATION, duration);
        intent.putExtra(IDef.EXTRA_POS, mCurPos);
        FooMonUtil.getInstance().postBroadcast(this, intent);
}
```

代码 9 中，播放服务将播放进度等信息填充到意向对象中并以广播的形式发送（给主 Activity。代码 10 是主 Activity 对收到播放进度广播的响应：

代码 10　主 Activity 响应播放进度广播

文件名：MainAct.java

```
class FeedbackReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context ctx, Intent intent) {
if(intent.getAction().compareTo(IDef.FEEDBACK_ACTION)! =0) {
                return;
            }
            //解析消息内容
            Bundle bundle = intent.getExtras();
            final int state = bundle.getInt(IDef.EXTRA_STATE);
                        final int progress = bundle.getInt(IDef.
EXTRA_PROGRESS);
                        final int length = bundle.getInt(IDef.
EXTRA_DURATION);
            final int pos = bundle.getInt(IDef.EXTRA_POS);
            //更新状态
        MainAct.this.updateState(state, progress, length, pos);
        }
};
```

代码 10 中，主 Activity 先解析消息内容，然后依据消息内容更新可视组件的状态（播放进度条、播放位置等）。

4.2.3 获取网络音乐资源信息

网络音乐资源信息的获取是以 HTTP 的方式发送请求，得到以 JSON 格式返回的信息数组。代码 11 是获取网络音乐资源信息的主要代码：

代码 11　获取网络音乐资源信息

文件名：InetResAct.java

```
        private ArrayList<MusicRes> getInetRes() throws
IOException, JSONException {
        //获取服务接口地址
        final String url = FooPrefsUtil.getInstance ().updateUrl
(this, IDef.Url_Res_Read);
        //请求参数
        Hashtable <String,String > params =  new Hashtable <
String,String>();
        params.put("category", "classic");//经典分类
        params.put("sub-category", "eu-us"); //欧美
        params.put("filter", "mp3,wma"); //资源类型
        //初始化容器
                ArrayList<MusicRes> items = new ArrayList<
MusicRes>();
        //发送请求并获取 JSON 数组
      JSONArray arr = FooHttpUtil.getInstance ().doPost2 (url,
params);
        if(arr!  =null) {
                for(int i = 0; i < arr.length(); ++i) {
                        JSONObject obj = arr.getJSONObject(i);
                        //把 JSON 对象转换成音乐资源对象
                        MusicRes res = new MusicRes();
                        res.setTile(obj.getString(MusicRes.Title));
                        res.setUri(obj.getString(MusicRes.Uri));
                        res.setSize(obj.getInt(MusicRes.Size));
                        items.add(res);
                }
        }
        return (items);
}
```

代码 11 中，互联网资源 Activity 调用 Http 工具类向服务端发送请求并获取包含网络音乐资源的 JSON 对象数组，再将 JSON 对象转换成音乐资源对象。代码 12 是 Http 工具类获取网络音乐资源信息的主要代码：

代码 12　Http 工具类获取网络音乐资源信息

文件名：FooHttpUtil.java

```
public JSONArray doPost2 (String url, Map<String, String> params) {
    StringBuffer sb = new StringBuffer();
    try {
        // 创建连接，发送请求并接受回应
        DefaultHttpClient client = new DefaultHttpClient();
        HttpPost post = new HttpPost(url);
        //添加请求参数
        if (params ! = null && params.size() > 0) {
        List<BasicNameValuePair> pairs = new ArrayList<BasicNameValuePair>(params.size());
            Set<String> keys = params.keySet();
            for (String key : keys) {
pairs.add(new BasicNameValuePair(key, params.get(key)));
            }
post.setEntity(new UrlEncodedFormEntity(pairs, "utf-8"));
        }
        HttpResponse resp = client.execute(post);
        //返回状态
    int statusCode = resp.getStatusLine().getStatusCode();
        if (statusCode == HttpStatus.SC_OK) {
            //获取回应条目（JSON 格式）
            HttpEntity entity = resp.getEntity();
            if(entity ! = null) {
                InputStream is = entity.getContent();
          BufferedReader br = new BufferedReader(new InputStreamReader(is));
                //读取服务端回复
                String line = null;
                while ((line=br.readLine())! =null) {
                    sb.append(line);
                }
                br.close();
            }
        }
        post.abort();
        //构建 JSON 数组
        JSONArray arr = new JSONArray(sb.toString());
        return (arr);
    } catch( ClientProtocolException e) { e.printStackTrace();
    } catch (IOException e) { e.printStackTrace();
    } catch(JSONException e) { e.printStackTrace();
    }
    return (null);
}
```

代码 12 中，Http 工具类使用 Http 客户端组件（HttpClient）执行 POST 请求，并获取客户端的回复，并将回复内容转换成 JSON 数组。

4.2.4 服务端音乐资源信息接口

代码 13 是服务端所提供的音乐资源信息接口定义，代码 12 中正是通过该接口地址（即参数 url）来获取网络音乐资源信息：

代码 13　服务端音乐资源信息接口

文件名：service.ws

```
<%
    final String path = request.getContextPath();
    final String basePath = request.getScheme () + "://" + request.getServerName()+":"+
                                request.getServerPort()+path+"/";
    final String baseRealPath = this.getServletContext ().getRealPath("/");
    //获取请求参数
final String cat = request.getParameter("category"); //大类
    final String sub_cat = request.getParameter ("sub-category"); //子类
final String filter = request.getParameter("filter"); //过滤条件
    System.out.println ("category="+cat+"|sub-category="+sub_cat+"|filter="+filter);
    //遍历本地（音乐资源服务器）音乐资源
    File dir = new File(baseRealPath+"res");
    if(! dir.exists()) { return; }
    //遍历文件夹中的文件
    File[] files = dir.listFiles();
    if(files.length < 1) { return; }
    //将资源信息包装成 JSON 格式返回给客户端
    JSONArray array = new JSONArray();
    for(int i = 0; i < files.length; ++i) {
        JSONObject obj = new JSONObject();
        obj.put("title", files[i].getName() ); //资源标题
obj.put("uri", basePath+"res/"+files[i].getName()); //资源 URL
        obj.put("size", files[i].length()); //资源大小
        array.put(obj);
    }
    //以 JSON 格式回复请求方
FooMiscUtil.getInstance().printAsJson(response, array);
%>
```

代码 13 中，信息接口通过遍历音乐资源服务器上的音乐资源文件，并将资源信息包装成 JSON 格式并返回给客户端。

4.2.5 音乐资源列表适配器

在界面设计中，音乐资源列表是该播放器应用最核心的展示组件，为了增强视觉效果，作者对奇偶行以及被选行都采用了不同的背景，而要实现该效果，则需要定制列表适配器。代码 14 是音乐资源列表适配器获取行视图的主要代码：

代码 14　音乐资源列表适配器主要代码

文件名：FooListAdapter.java

```
@Override
public View getView (int pos, View convertView, ViewGroup parent) {
    if(convertView == null) {
    final LayoutInflater inflater = this.ctx.getLayoutInflater();
```

```
            //获取项目 View(每一行)
        convertView = inflater.inflate (R.layout.file_item_view,
null, false);
        }
        if(pos == selectedPos) { //设置选择行的背景
  convertView.setBackgroundResource(R.drawable.lorange);
        } else {
            if((pos%2)>0) { //奇数行的背景
        convertView.setBackgroundResource(R.drawable.gray);
            } else { //偶数的背景
        convertView.setBackgroundResource(R.drawable.silver);
            }
        }
        //设置显示内容
        ((TextView)convertView.findViewById (R.id.tv_index)).
setText(""+(pos+1));
        ((TextView)convertView.findViewById (R.id.tv_content)).
setText(items.get(pos));
        return (convertView);
}
```

代码 14 中，在基本适配器的重载函数“getView”中，通过行的属性信息（行号及是否被选）来设置不同类型行的显示背景。

4.2.6 网络音乐资源下载

对于网络音乐资源，其 URI 就是完整的 HTTP 路径，所以使用 HttpURLConnection 即可实现该资源的下载，代码 15 是下载网络音乐资源的主要代码：

代码 15　下载网络音乐资源

文件名：DownloadThread.java

```
@Override
public void run() {
        //生成保存路径
        final String filePath = FooPrefsUtil.getInstance ().
getString(ctx, IDef.EXTRA_OUT);
        final File f = new File(filePath);
        final String destDir = f.getParentFile().getAbsolutePath();
        final String uri = res.getUri();
        final int size = res.getSize();
        //获取资源 URI 中的文件名(需要编码)和上下文路径(无
//需编码)
        final String filename = FooFileUtil.getInstance ().
getFilename(uri);
        final String context = uri.replace(filename, "");
        final String encodeUri = context +URLEncoder.encode
(filename);
        File dir = new File(destDir);
        if(! dir.exists()) { return; }
        File destFile = new File (destDir +File.separator +
filename);
        //发送通知
        postNotification("开始下载"+filename);
        int abyte = -1;
        int index = 0;
        int step = 1;
        try {
            URL url = new URL(encodeUri);
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            InputStream is = conn.getInputStream();
FileOutputStream fos = new FileOutputStream(destFile);
            //计算下载进度检查点(10%)
            final int pos = (int)(size*0.1f*step);
            while((abyte=is.read())! =-1) {
                fos.write(abyte);
                index++;
                //以通知的方式更新下载进度信息
                if(index%pos==0) {
            postNotification(filename+"已下载"+step+"0%");
                    step++;
                }
            }
            is.close();
            fos.close();
            conn.disconnect();
            //发送通知
            postNotification(filename+"下载完成！");
        } catch(IOException e) { e.printStackTrace(); }
}
```

在代码 15 中，下载线程通过系统通知机制提示下载进度信息，其效果如图 4 所示（系统状态栏显示了加载进度，以 10%为单位）。

5　结语

限于篇幅，对该音乐资源播放器的部分功能细节并没有给予详细的介绍（例如：侦听电话呼入状态并进行播放的控制、播放列表的生成、使用文件浏览器来设置下载目录等），读者可以通过配套代码进行理解。

该开发实例涵盖了移动客户端应用中数据展示的常见模式，例如：网络音乐资源的获取、定制的音乐资源列表适配器等；对 Android 平台组件交互的介绍非常实用，例如：广播消息机制、系统消息机制（还有线程消息队列机制），都是在商业项目开发中常用的方式。

该开发案例经 Android 2.2.1、2.3.6 和 4.0.4 版本的实机验证，希望对读者有所参考。

（收稿日期：2013-03-29）

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现

17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码

19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码

15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用

2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计

10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现