



山东大学 2012 年度软件学院学生科技创新项目

**NewRadio**

设计文档

邵宗瑞、王立鹏、韩俊晓、魏文静

山东大学软件学院

### 特性:

1. podcast 信息动态更新，推送功能开发中。
2. about 页面内容与服务器自动同步，方便广告投放与内容修改。
3. 信息通信使用 xml 技术，节省流量，加快速度，维护方便。
4. 断点续传及缓冲技术，方便用户在复杂移动网络中使用。
5. 哼唱搜索 技术，用户只要轻轻哼唱旋律，程序就会从服务器中迅速找到相应节目，方便用户再次收听。
6. 自动更新技术，使应用程序永远保持年轻活力。
7. 用户信息统计，方便节目作者挖掘用户习惯，优化节目内容。
8. 社交网络分享功能开发中。

### 简介:

**播客**，亦常直接称作“**Podcasting**”，是指一种在互联网上发布文件并允许用户订阅 feed 以自动接收新文件的方法，或用此方法来制作的电台节目。

NewRadio 原型是 IOS 上的一个程序，里面集合了 6 位著名电台主持人的 podcasting，因为我比较喜欢其中几位的节目，所以比较关注这个应用的发展。偶然的机会，我得知这款应用没有 Android 版本，于是萌生想法，想自己做一款 Android 上的 NewRadio，于是诞生了此项目。



首先，我将 ipod touch 连接到代理服务器上，打开 IOS 上的 NewRadio，然后监控代理服务器的记录，由此找出了 IOS 版本 NewRadio 所用的网络接口。如下：

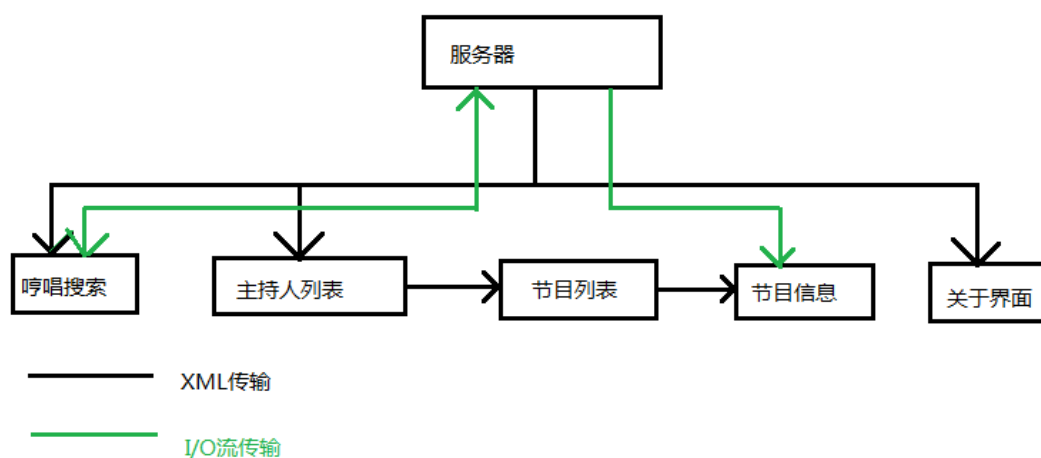
```

19 CONNECT p10-calDav.icloud.com:443 - DIRECT/17.151.224
2 CONNECT p10-calDav.icloud.com:443 - DIRECT/17.151.224
2 CONNECT p10-calDav.icloud.com:443 - DIRECT/17.151.224
0194 GET http://42.121.19.82/newradio/Podcast.aspx - DI
37 GET http://42.121.19.82/newradio/DJ.aspx - DIRECT/42
00 GET http://42.121.19.82/newradio/Album.aspx - DIRECT
00 GET http://42.121.19.82/newradio/Album.aspx - DIRECT
0194 GET http://42.121.19.82/newradio/Podcast.aspx - DI

```

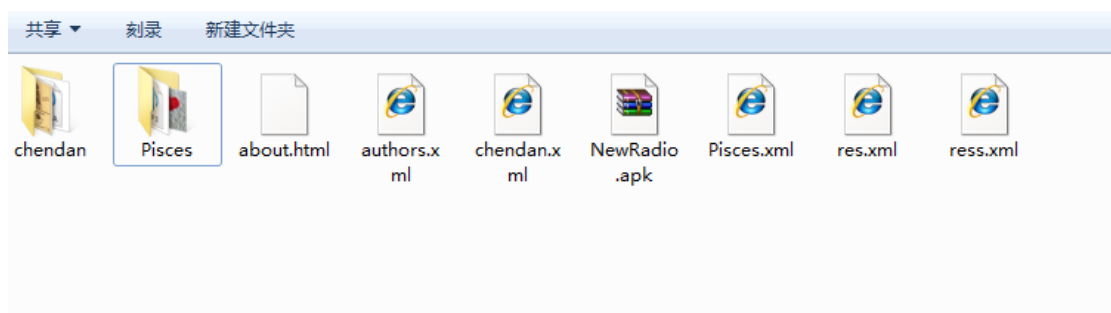
经过查看网页源代码，发现客户端与服务器之间采用 xml 通讯，也就是所有信息存储在服务器上的 xml 文件中，客户端获取 xml 文件后进行相应的解析，展现给用户。然后用户根据 xml 中的信息点击下载相应文件。

大体方向确定后，整体的架构如下：

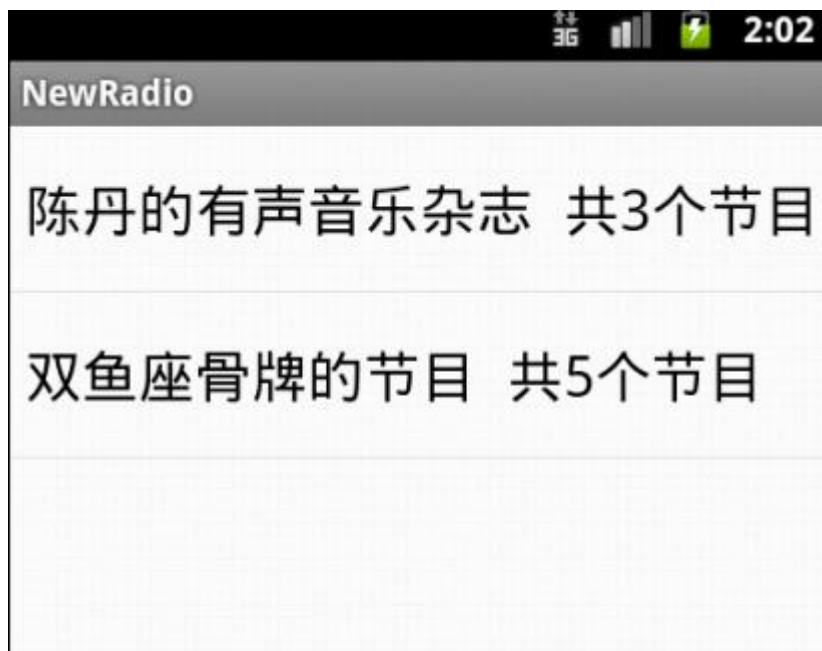


详解：

1. 服务器上存储 XML（包含主持人和节目的信息），存储节目资源（图片和音频）。



2. 启动客户端，客户端从服务器下载 xml 文件后解析，展现相应信息给用户（主持人信息、节目信息等）。



3. 用户点击具体节目，选择相应功能（下载、播放、删除等），客户端根据用户操作来执行相应操作（下载节目、播放节目、删除节目等）。



4. 另外，利用现有平台（umeng 和盛大哼唱搜索），增加了自动更新、信息统计、哼唱搜索等功能。



5. Xml 解析使用 android 的 pull 方法，该方法效率高，可定制更新（更新过程可人为中断，

只解析更新部分，是 Google 推荐的方法)：

```
public class PullAuthorHandler {

    //解析用到的tag
    private String _entryname = "author";
    private String _Cname = "Cname";
    private String _Ename = "Ename";
    private String _program_name = "program_name";
    private String _summary = "summary";
    private String _tot_program = "tot_program";

    //用于保存xml解析获取的结果
    private ArrayList<AuthorEntry> authorEntryList = null;
    private AuthorEntry authorEntry = null;
    private Boolean startEntryElementFlag = false;

    //解析xml数据
    public void parseXML(String xml) {
        //解析xml数据
        Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(xml);
        Element root = doc.getDocumentElement();
        NodeList nList = root.getElementsByTagName("author");
        for (int i = 0; i < nList.getLength(); i++) {
            Element e = nList.item(i);
            if (e.getNodeName().equals("author")) {
                AuthorEntry authorEntry = new AuthorEntry();
                authorEntryList.add(authorEntry);
                authorEntry = null;
                startEntryElementFlag = false;
            }
        }
    }
}

public class PullProgramHandler {

    //解析用到的tag
    private String _entryname = "program";
    private String _entryID = "id";
    private String _entrytile = "title";
    private String _entrysubtitle = "subtitle";
    private String _entrypushtime = "pushtime";
    private String _entryhow_long = "how_long";
    private String _entrycomment = "comment";
    private String _entryimg = "img";
    private String _entrysource = "source";
    private String _author = "author";
    private String _filesize = "filesize";

    //用于保存xml解析获取的结果
    private ArrayList<ProgramEntry> programEntryList = null;
    private ProgramEntry programEntry = null;
    private Boolean startEntryElementFlag = false;

    //解析xml数据
    public void parseXML(String xml) {
        //解析xml数据
        Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(xml);
        Element root = doc.getDocumentElement();
        NodeList nList = root.getElementsByTagName("program");
        for (int i = 0; i < nList.getLength(); i++) {
            Element e = nList.item(i);
            if (e.getNodeName().equals("program")) {
                ProgramEntry programEntry = new ProgramEntry();
                programEntryList.add(programEntry);
                programEntry = null;
                startEntryElementFlag = false;
            }
        }
    }
}
```

6. 程序实现了一个下载类，可以实现下载、获得网络上文件大小、断点续传的功能：

```
public class Downloader {

    //下载状态
    private final int STOP = -1;
    private final int DOWN = 1;
    private int state;

    //定义的一些常量变量，看名字就知道什么意思了
    private static final int BUFFER_SIZE = 1024;
    private String urlStr;
    private String savePath;
    private int downloadPercent = 0;
    private boolean completed = false;

    private Handler handler;

    public Downloader(String urlStr, String savePath, Handler handler) {
        this.urlStr = urlStr;
        this.savePath = savePath;
        this.handler = handler;
    }

    public void startDownload() {
        state = STOP;
        download(urlStr, savePath);
    }

    private void download(String urlStr, String savePath) {
        try {
            URL url = new URL(urlStr);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            conn.connect();
            int responseCode = conn.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                state = DOWN;
                long contentLength = conn.getContentLength();
                File file = new File(savePath);
                if (!file.exists()) {
                    file.createNewFile();
                }
                FileOutputStream fos = new FileOutputStream(file);
                BufferedOutputStream bos = new BufferedOutputStream(fos);
                byte[] buffer = new byte[BUFFER_SIZE];
                int len;
                while ((len = conn.getInputStream().read(buffer)) != -1) {
                    bos.write(buffer, 0, len);
                    downloadPercent += len * 100 / contentLength;
                    handler.updateProgress(downloadPercent);
                    if (downloadPercent == 100) {
                        completed = true;
                        state = STOP;
                        handler.finish();
                    }
                }
                bos.flush();
                bos.close();
                fos.close();
            } else {
                state = STOP;
                handler.finish();
            }
        } catch (Exception e) {
            state = STOP;
            handler.finish();
        }
    }
}
```



7.程序实现的 about 页面基于 html 解析技术,可动态更新里面的内容,每次展现 about 信息,客户端都判断服务器上的 about 信息是否更改,如果更改过则下载新的 about 页面进行展示,方便广告投放 和 about 信息更改。

```
public class About extends Activity{
    WebView web = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);
        MobclickAgent.onError(this);
        MobclickAgent.updateOnlineConfig(this);
        MobclickAgent.setDefaultReportPolicy(this, ReportPolicy.BATCH_AT_LAUNCH);

        web = (WebView) findViewById(R.id.webView1);

        File f = new File(BASE.basePath+"about.html");
        Handler handler = new Handler(){
            public void handleMessage(Message msg){
                if(msg.what==1)
                {
                    web.loadUrl("file://" + BASE.basePath + "about.html");
                }
                else if(msg.what==2)
                {
                    Toast.makeText(About.this, "网络连接失败!", Toast.LENGTH_SHORT).show();
                }
            }
        };
        if(!f.exists() || f.length() != Downloader.getFileSize(BASE.baseUrl + "about.html", handler))
        {
            if(f.exists()) f.delete();
            Downloader down = new Downloader(BASE.baseUrl + "about.html", BASE.basePath + "about.html", handler);
            down.start();
        }
        else
    }
```

## 8.哼唱搜索

**哼唱搜索** 是近些年来发展起来的一项新的技术,各大公司 (google、baidu、apple、盛大) 等都有自己的引擎, **哼唱搜索** 允许用户在哼唱节目旋律后找到相应节目,方便用户查找歌曲节目、语音搜索互动等。

本程序使用的哼唱搜索引擎源自 盛大科学院 (请勿商业使用)。搜索功能在服务器上进行,搜索完成后,服务器将结果返回给客户端进行显示。



## 哼唱搜索

开始录音

录制时间：0s

3. 爱情故事

2. Say you Say me

1. 麦兜

```

Thread t = new Thread(new Runnable() {

    @Override
    public void run() {
        // TODO Auto-generated method stub
        try {

            Socket soc = new Socket("211.87.227.116", 8998);
            File f = new File(AudioName);

            FileInputStream fin = new FileInputStream(f);
            OutputStream out = soc.getOutputStream();
            int bit = fin.read();
            while(bit!=-1)
            {
                out.write(bit);
                bit = fin.read();
            }

            out.close();
            fin.close();
            soc.close();
            Message msg = Message.obtain();
            msg.what=2;
            sendOk.sendMessage(msg);

        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

```

public void run() {
    // TODO Auto-generated method stub

    try {
        System.out.println("start\n");
        File f = new File("G:\\workspace_win\\SDHumming\\humming\\tmp.mp4");
        if(f.exists()) f.delete();
        InputStream fin= ss.getInputStream();
        FileOutputStream out = new FileOutputStream(f);

        int bit = fin.read();
        while(bit!=-1)
        {
            out.write(bit);
            bit = fin.read();
        }

        fin.close();
        out.close();
        ss.close();
        System.out.println("Receieve OK\n");

        File fout = new File("G:\\workspace_win\\SDHumming\\humming\\out.wav");
        if(fout.exists()) fout.delete();
        fout = new File("G:\\workspace_win\\SDHumming\\humming\\res.txt");
        if(fout.exists()) fout.delete();

        Runtime.getRuntime().exec("G:\\workspace_win\\SDHumming\\humming\\mp4towav.bat");
        Runtime.getRuntime().exec("G:\\workspace_win\\SDHumming\\humming\\run.bat");
    }
}

```