计算机学院2014级学年论文

# **完 美 天 气**

专业名称 软件工程

2017 年 8 月 29 日

# 完 美 天 气 A P P

内容摘要 在智能手机中，Android平台约占75％市场份额。安卓系统的广泛应用，使得Android手机应用程序的数量快速增长。Android APP也逐渐成为人们生活不可或缺的一部分。本设计开发了一种运用在Android系统上的手机天气预报软件系统，本系统通过选择城市来获得天气，风向，温度，生活健康提示等信息，以及备忘录日历等功能。基于手机的该软件极大的方便了用户的出行和行程安排，避免了不必要的麻烦，具有很强的实用性。

关键词 Android，天气预报APP，完美天气

1.1 章 需 求 分 析

1.1.1 开发背景

随着智能手机的不断普及，移动应用开发成为当下最热门的技术之一。在 Google 和 Android 手机联盟的共同推动下，Android 在众多移动应用开发平台中脱颖而出。Android 是一个真正意义上的开源智能手机操作系统，该系统一经推出立即受到全球移动设备厂商和开发者的热捧。为顺应潮流，本设计旨在搭载 Android 的移动设备上运行，方便用户的出行和行程安排。

* + 1. 项目需求分析

根据功能的需求，主要功能应具备如下几点：

1. 定位查询全国各地城市未来几天内的天气状况

2. 给出实时的出行建议和生活指数建议

3. 系统要实现三级联动查询城市

4. 系统实现备忘录和日历功能

2.1 章 总 体 设 计

2.1.1 系统规划

由上述的需求，现将系统分为三大模块：天气显示模块、城市切换模块、日历模块与备忘录模块。各系统模块功能如下：

1). 天气显示模块

显示指定城市三天内的天气状况，包括时间、城市名称、温度、湿度、风力与当日的各项指数，用户可通过按定位键来进行定位显示当前城市相关指数，以及跳转至设置预报城市界面来更换城市。

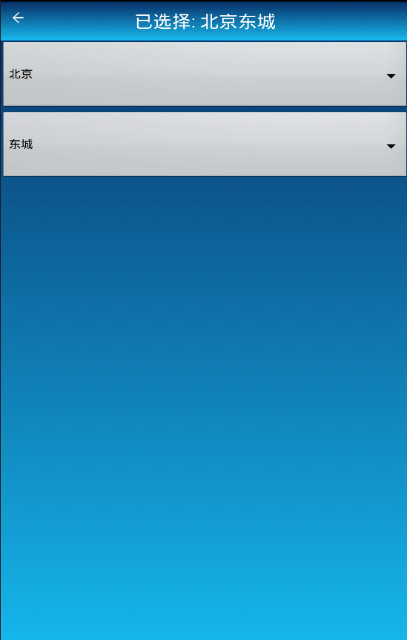
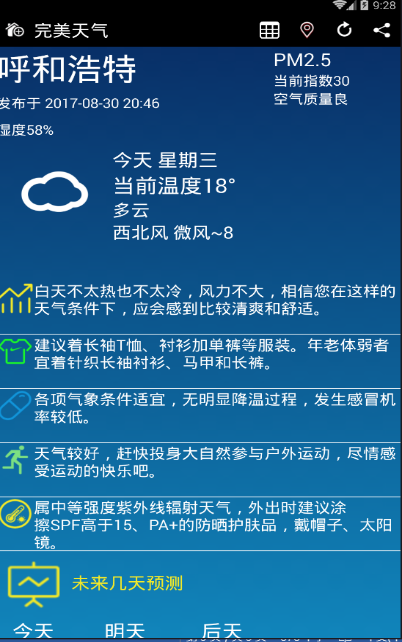
2). 预报城市设置模块

由自动设置预报城市与手动设置二部分组成，自动设置实现 GPS 定位功能，自动确定当前用户所在地；而手动设置则通过三级联动选择省，城市，地区。

3). 备忘录日历模块

进入日历模块以后，可以查看相应的日历。点击相应日历可以添加对应的备忘录事件，也可以进入单独备忘录列表进行管理备忘录

2.1.2 功能界面





3.1 章 系 统 设 计

3.1.1 开发及运行环境

JDK1.6 adt-bundle-windows-x86\_64-20130219

SDK17 夜神模拟器Android4.4.2

3.1.2 数据库设计

两张表：

CITY表 存放城市信息，用于保存最近一次使用的城市信息，下次打开APP直接显示该城市对应信息。

MEMEORY表 备忘录表 存放备忘录内容 时间

3.1.3 主要流程

1. 搭建相应Android环境，建立命名为pfweather项目

2. 进入welcome欢迎界面

3. 程序首次运行，建立相应数据库和表，将默认城市呼和浩特进行显示

4．城市选择界面，解析city.JSON 获取相应的省份，城市，地区对应关系

5. 若选择定位，则调用高德API进行定位并将城市名字传入天气API串

6. 将城市名字传入和风天气对应API串中，进行请求返回对应JSON串进行解析显示到对应控件上

7.若打开日历备忘录界面，则前台进行操作对应后台数据库进行增删查改。

3.1.4 主要方法技术

1. Android界面 UI 设计 2. Android 的网络通信

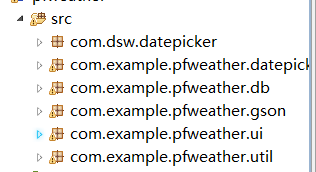
3. Android 的广播机制 4. 高德API的调用实现定位

5. JSON 手动解析 与 GSON工具解析

6. SQLite 数据库相应操作 7. 调用和风天气API获取天气数据

4.1 章 具 体 实 现

4.1.1 项目结构



4.1.2 主要功能实现

4.1.2.1 手动解析JSON 获取城市列表

先将Json串转为String串

/ 初始化读取city.json为String

**private** String initData() {

StringBuffer sb = **new** StringBuffer();

AssetManager assetmanager = **this**.getAssets();

**try** {

InputStream is = assetmanager.open("city.json");

**byte**[] data = **new** **byte**[is.available()];

**int** len = -1;

**while** ((len = is.read(data)) != -1) {

sb.append(**new** String(data, 0, len, "gb2312"));

}

is.close();

**return** sb.toString();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** **null**;

}

再手动解析JSON

JSONObject object = **new** JSONObject(cityJson);

JSONArray array = object.getJSONArray("citylist");

sprovience = **new** String[array.length()];

String mprovience = **null**;

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

JSONObject mProvienceObject = array.getJSONObject(i);

**if** (mProvienceObject.has("p")) {

mprovience =mProvienceObject.getString("p");//取省份具体值

provience[i] = mprovience;

} **else** {

sprovience[i] = "未知地点";

}

其他类似，逐层解析出省份，城市，地区并存在对应String数组中

通过Map将省-市；市-区 进行关联；利用Spinner 将对应信息绑定显示

// 存省对应的所有市

**private** Map<String, String[]> mCitiesDataMap = **new** HashMap<String, String[]>();

// 存市对应的所有区

**private** Map<String, String[]> mAreaDataMap = **new** HashMap<String, String[]>();

mAreaDataMap.put(mcity, sarea);

mCitiesDataMap.put(mprovience, scity);

4.1.2.2 天气实体类获取

Weather实体类的获取：

此处传入城市名字，通过从网上请求下载对应Json串转为String串之后进行操作

// 通过城市获取对应Weather实体类

**public** **static** Weather UpdateWeather(String city) {

String cname; String MyKey = "6130363df20b4d1ea928bb07257ace9a";

cname = URLEncoder.*encode*(city, "utf-8");

String url = "https://free-api.heweather.com/v5/weather?city="+ cname + "&key=" + MyKey; String jsonStr = UpdateUtil.*getWeString*(url);

JsonParser parse = **new** JsonParser();

JsonObject jsonObject = (JsonObject) parse.parse(jsonStr);

JsonArray heWeather = jsonObject.get("HeWeather5").getAsJsonArray();

**if** (heWeather.size() == 0) {

**return** **null**;

} **else** {String weatherStr = heWeather.get(0).toString();

Weather weather = **new** Gson().fromJson(weatherStr, Weather.**class**);

**return** weather;

因为用到网络通信，所以需要新起一个线程利用handle进行传递消息

Handler handler = **new** Handler() {

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

**super**.handleMessage(msg);

Bundle data = msg.getData();

Weather w = (Weather) data.getSerializable("value");

citydb.city\_insert(cname); updateView(w); }};

Runnable runnable = **new** Runnable() {

**public** **void** run() {Message msg = **new** Message();

Bundle data = **new** Bundle();

data.putSerializable("value", UpdateUtil.*UpdateWeather*(cname));

msg.setData(data); handler.sendMessage(msg); }};

4.1.2.3 定位模块实现

调用高德API 首先在AndroidMainfest中定义相关权限和KEY

<uses-permission android:name=*"android.permission.INTERNET"* />

<uses-permission android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"* />

<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"* />

<uses-permission android:name=*"android.permission.ACCESS\_WIFI\_STATE"* />

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"* />

<uses-permission android:name=*"android.permission.CHANGE\_WIFI\_STATE"* />

<uses-permission android:name=*"android.permission.READ\_PHONE\_STATE"* />

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />

<uses-permission android:name=*"android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"* />

<uses-permission android:name=*"android.permission.WAKE\_LOCK"* />

</meta-data>

<service android:name=*"com.amap.api.location.APSService"*></service>

//初始化定位函数**public** **void** initlocate() {// 启动定位

mLocationClient = **new** AMapLocationClient(getApplicationContext());

mLocationClient.setLocationListener(**this**);

AMapLocationClientOption mLocationOption = **new** AMapLocationClientOption();

mLocationOption.setLocationMode(AMapLocationMode.*Hight\_Accuracy*);

mLocationOption.setInterval(2000);

mLocationOption.setOnceLocation(**true**);

mLocationClient.setLocationOption(mLocationOption);

mLocationClient.startLocation();}

//定位回调函数，手动进行实现

**public** **void** onLocationChanged(AMapLocation location) {

cname = location.getCity();

**new** Thread(runnable).start();

Weather newweather = UpdateUtil.*UpdateWeather*(location.getCity());

updateView(newweather);

mLocationClient.stopLocation();}

4.1.2.4 每次打开APP展示最近选择城市

Handler handler = **new** Handler() {**public** **void** handleMessage(Message msg) {

**super**.handleMessage(msg); Bundle data = msg.getData();

Weather w = (Weather) data.getSerializable("value");

citydb.city\_insert(cname); updateView(w);

}};

每次更新城市时候将名字传入数据库中 打开APP时候 从数据库表中取得最近一次的城市名字

Cursor cursor = citydb.select("city\_table");

**if** (cursor.moveToLast()) {cursor.moveToLast();

cname = cursor.getString(cursor.getColumnIndex("city\_name"));}

4.1.2.5 备忘录列表实现

从数据库读取并进行刷新显示

**public** **void** RefreshNotesList() {

// 如果dataList已经有的内容，全部删掉 // 并且更新simp\_adapter

**int** size = dataList.size();**if** (size > 0) {

dataList.removeAll(dataList); simple\_adapter.notifyDataSetChanged();

}// 从数据库读取信息

Cursor cursor=memorydb.select("memory\_table");

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

String name = cursor.getString(cursor.getColumnIndex("memeory\_content"));

String date = cursor.getString(cursor.getColumnIndex("memeory\_date"));

Map<String, Object> map = **new** HashMap<String, Object>();

map.put("tv\_content", name); map.put("tv\_date", date); //放于TextView

dataList.add(map); }

simple\_adapter = **new** SimpleAdapter(**this**, dataList, R.layout.*mem\_item*,

**new** String[] { "tv\_content", "tv\_date" }, **new** **int**[] {

R.id.*tv\_content*, R.id.*tv\_date* });listview.setAdapter(simple\_adapter);}

4.1.2.6 备忘录编辑实现

通过标志位区分

**public** **int** enter\_state = 0;// 0是新建一个备忘录，1是修改已有备忘录

SQLiteDatabase db = memorydb.getReadableDatabase();

// 获取edittext内容 String content = et\_content.getText().toString();

// 标志为0---添加一个新的记录 **if** (enter\_state == 0) {

**if** (!content.equals("")) {// 获取此时时刻时间

Date date = **new** Date();SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy MM dd");

String dateString = sdf.format(date); // 向数据库添加信息

ContentValues values = **new** ContentValues();values.put("memeory\_content", content);

values.put("memeory\_date", dateString); db.insert("memory\_table", **null**, values);

finish();}

**else** {Toast.*makeText*(noteEditActivity.**this**, "请输入你的内容！",

Toast.*LENGTH\_SHORT*).show();}}

**else** {// 标志位1--查看并修改一个已有的日志

ContentValues values = **new** ContentValues();values.put("memeory\_content", content);

db.update("memory\_table", values, "memeory\_content = ?",

**new** String[] { last\_content });finish();}

4.1.2.7 备忘录长按和短按实现不同效果

// 点击listview中某一项item的点击监听事件

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2, **long** arg3) {String content = listview.getItemAtPosition(arg2) + "";

String content1 = content.substring(content.indexOf("=") + 1,content.indexOf(","));//截取字符串content内容

Intent myIntent = **new** Intent(NoteActivity.**this**, noteEditActivity.**class**);

Bundle bundle = **new** Bundle();

bundle.putString("info", content1);

bundle.putInt("enter\_state", 1);//状态变为修改

myIntent.putExtras(bundle);

startActivity(myIntent); }

// 点击listview中某一项长时间的点击事件

@Override

**public** **boolean** onItemLongClick(AdapterView<?> arg0, View arg1,**final** **int** arg2, **long** arg3) {Builder builder = **new** Builder(**this**);

builder.setTitle("删除该备忘录");

builder.setMessage("确认删除吗？");

builder.setPositiveButton("确定", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** which) {

String content = listview.getItemAtPosition(arg2) + "";

String content1 = content.substring(content.indexOf("=") + 1,

content.indexOf(","));

DB.delete("memory\_table", "memeory\_content = ?", **new** String[] { content1 });

RefreshNotesList();//删除后进去刷新

}

});

builder.setNegativeButton("取消", **new** DialogInterface.OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** which) {

}

});builder.create();builder.show();**return** **true**; }

4.1.2.8 日历实现

我们要实现的日历控件采用ViewPager作为主框架，CalendarView继承ViewPager，这样就天生拥有左右滑动和缓存的功能。目前我们设定日历左右滑动为月份切换的操作，每一个月份显示通过自定义ViewGroup实现，也就是我们的MonthView，月份中的日期是通过layout布局解析出的View，根据月份的不同每个MonthView可能包含6 x 7或5 x 7个日期View，由于给ViewPager绑定数据需要通过PagerAdapter，所以继承PagerAdapter我们扩展了一个CalendarPagerAdapter，来完成MonthView的相关初始化和日期数据的绑定。每个MonthView的日期数据应该由上个月的后0~6天、当前月的天数和下个月的前0~6天组成。首先计算出当前月有多少天，这个简单，以及根据年月算出当前月的第一天是星期几：返回0代表周日，1~6代表周一到周六，总页数应由日历的起始年月得到，其实就是确定ViewPager的总页数MonthView继承ViewGroup，也就是日历的每一页，接收到日期数据后，在MonthView中根据数据构造对应的日期View，然后添加View到MonthView中，最后通过**onMeasure**、**onLayout**确定每个View最终大小和位置。

5.1 章 软 件 测 试

5.1.1 测试目的

(1) 确认软件的质量，确保软件能正常运行，并且达到了期待中的效果

(2) 确认信息的正确性。确保从网上下载的天气情况与实际的天气情况没有什么差别



6.1 章 总 结 与 展 望

本系统基本实现了需求的中的天气预报的功能，界面设计较漂亮，具有一定的实用性。其中的天气数据全部来源于和风天气API，定位基于高德API，故在运行时一定要确保系统所处环境的网络流畅性。

由于开发基于夜神模拟器Android4.4.2 测试结果有一定局限性，可能存在由于手机分辨率不同导致页面无法显示预想的效果。

但是由于时间有限，所以在整个软件中的内容还有一些不太完善，还存在着一些尚待提高和改进的地方。希望能在以后的学习中能够不断的完善和改进。如可以将备忘录部分按时间的重要性进行分类排序，并可以增加对备忘录的属性分类等功能。

参考文献

1. 高德定位SDK开发文档

http://lbs.amap.com/api/android-location-sdk/locationsummary/

1. 和风API接口说明文档 <https://www.heweather.com/documents/api/v5>
2. JSON解析中的手动和使用GSON解析

<http://www.cnblogs.com/zhujiabin/p/5498272.html>

【4】 日历实现

http://www.jianshu.com/p/304c8e70d0bd

要不是毕业好几年我都不舍得分享出来！！！

CTRL+D收藏一下或者关注走一波-有你所需！不断更新！

其他相关下载，配套代码以及PPT。稳妥的小老弟

https://me.csdn.net/download/qq\_27500493