

## 奋斗的菜鸟ing

每天都要学习，每天都要有进步。

目录视图

摘要视图



18 订阅

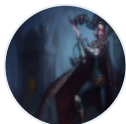


举报



关闭

### 个人资料



zpengyong



博客专家

原创

123

粉丝

323

喜欢

210

评论

441



等级：博客 6

访问量：74万+

积分：7122

排名：3858

## [置顶] Android WiFi开发（一）扫描、连接、信息

标签：wi-fi android 重复 扫描 连接

2016年03月04日 13:56:15

49362人阅读

评论(68)

收藏

分类：

Android-wifi ( 3 )

版权声明：本文出自张朋永的博客，转载必须注明出处。  
<http://blog.csdn.net/VNanyesheshou/article/details/50771698>

目录(?)

[+]

在平常开发中经常会涉及到wifi相关的内容，在此将之前所用到的整理

操作wifi的相关类，主要放在android.net.wifi包下面。使用wifi相关方法

- ACCESS\_WIFI\_STATE
- CHANGE\_WIFI\_STATE
- CHANGE\_WIFI\_MULTICAST\_STATE

ACCESS\_WIFI\_STATE 获取WiFi状态。



### 少儿编程





统计

站长统计

博客专栏



Object-c基础  
文章：15篇  
阅读：15959



Android蓝牙  
文章：11篇  
阅读：189366

文章分类

- Android 蓝牙 (11)
- Android-wifi (4)
- Android 基础 (20)

CHANGE\_WIFI\_STATE 改变WiFi状态。

CHANGE\_WIFI\_MULTICAST\_STATE 改变WiFi多播状态

申请权限方式：在AndroidManifest.xml文件中填写

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

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



wifi相关操作所需要用到的类。

ScanResult	Describes information about a detected access point.
WifiConfiguration	A class representing a configured Wi-Fi network, including the security configuration.
WifiConfiguration.AuthAlgorithm	Recognized IEEE 802.11 authentication algorithms.
WifiConfiguration.GroupCipher	Recognized group ciphers.
WifiConfiguration.KeyMgmt	Recognized key management schemes.
WifiConfiguration.PairwiseCipher	Recognized pairwise ciphers for WPA.
WifiConfiguration.Protocol	Recognized security protocols.
WifiConfiguration.Status	Possible status of a network configuration.
WifiInfo	Describes the state of any Wifi connection
WifiManager	This class provides the primary API for managing Wi-Fi.
WifiManager.MulticastLock	Allows an application to receive Wifi Multicast traffic.
WifiManager.WifiLock	Allows an application to keep the Wi-Fi radio on.
WpsInfo	A class representing Wi-Fi Protected Setup (WPS).

先说一下wifi的基本使用情况。

关闭



少儿编程



[Android 多线程](#) (5)  
[Android 多媒体](#) (4)  
[Android 进阶](#) (8)  
[Android 遇见的问题](#) (4)  
[Android 系统源码](#) (3)  
[Android 测试](#) (2)  
[java基础](#) (10)  
[java关键字](#) (2)  
[iOS开发](#) (28)  
[Object-C开发入门](#) (15)  
[Effective Objective-C](#) (4)  
[C语言基础](#) (18)  
[Red5](#) (3)  
[FFmpeg](#) (2)  
[ubunt](#) (4)  
[OpenCV](#) (1)

## 文章存档

[2018年1月](#) (2)  
[2017年9月](#) (2)  
[2017年8月](#) (2)  
[2017年7月](#) (10)  
[2017年6月](#) (5)

展开

## 阅读排行

[objc]

```

1.  import java.util.List;
2.  import android.content.Context;
3.  import android.net.wifi.ScanResult;
4.  import android.net.wifi.WifiConfiguration;
5.  import android.net.wifi.WifiInfo;
6.  import android.net.wifi.WifiManager;
7.  import android.net.wifi.WifiManager.WifiLock;
8.  import android.widget.Toast;
9.
10. public class WifiAdmin {
11.     // 定义WifiManager对象
12.     private WifiManager mWifiManager;
13.     // 定义WifiInfo对象
14.     private WifiInfo mWifiInfo;
15.     // 扫描出的网络连接列表
16.     private List<ScanResult> mWifiList;
17.     // 网络连接列表
18.     private List<WifiConfiguration> mWifiConfiguration;
19.     // 定义一个WifiLock
20.     WifiLock mWifiLock;
21.
22.     // 构造器
23.     public WifiAdmin(Context context) {
24.         // 取得WifiManager对象
25.         mWifiManager = (WifiManager) context
26.             .getSystemService(Context.WIFI_SERVICE);
27.         // 取得WifiInfo对象
28.         mWifiInfo = mWifiManager.getConnectionInfo();
29.     }
30.
31.     // 打开WIFI
32.     public void openWifi(Context context) {
33.         if (!mWifiManager.isWifiEnabled()) {
34.             mWifiManager.setWifiEnabled(true);
35.         } else if (mWifiManager.getWifiState() == 2) {
36.             Toast.makeText(context, "亲, Wifi正在开启",
37.                 Toast.LENGTH_SHORT).show();
38.         } else {
39.             Toast.makeText(context, "Wifi已经开启",
40.                 Toast.LENGTH_SHORT).show();
41.         }
42.     }
43. }
  
```



18



关闭



少儿编程



Android WiFi开发 (一) 扫描...	(49253)
Android 蓝牙开发 (一) 蓝牙...	(41556)
FFmpeg 音频编码 (PCM数据...	(24825)
Android模拟位置信息	(21612)
Android 蓝牙开发 (二) Ble ...	(19976)
iOS UIActivityIndicator用...	(18315)
Android 蓝牙开发 (四) OPP...	(16656)
Android界面背景图片不显示----	(15610)
iOS ShareSDK实现分享——微...	(15380)
Android WiFi开发 (二) Wifi...	(15323)

### 最新评论

Android TCP socke...  
kin20180306 : 楼主能发到我邮箱吗1632634565@qq.com 谢谢

Android 蓝牙开发 (八) hf...  
zpengyong : [reply]aiTCR[/reply] 厉害了

Android 蓝牙开发 (二) Bl...  
zpengyong : [reply]aiTCR[/reply]  
??????什么情况。

Android 蓝牙开发 (八) hf...  
TTcccCarrie : 感谢分享了些我不熟悉的模块, 我还会再回来的。

Android 蓝牙开发 (八) hf...  
TTcccCarrie : 拨打, 挂断电话行为对应响应的方法。

Android 蓝牙开发 (八) hf...  
TTcccCarrie : 蓝牙耳机可以控制手机接听、拒接、挂断电话, 拨打电话等功能。

Android 蓝牙开发 (七) hf...  
TTcccCarrie : 蓝牙通话状态下, 切换到听筒、扬声器或者停止通话, 都会断开连接。

```

38.         Toast.makeText(context, "亲, Wifi已经开启, 不用再开了", Toast.LENGTH_SHORT).sh
        ow();
39.     }
40. }
41.
42. // 关闭WIFI
43. public void closeWifi(Context context) {
44.     if (mWifiManager.isWifiEnabled()) {
45.         mWifiManager.setWifiEnabled(false);
46.     } else if (mWifiManager.getWifiState() == 1) {
47.         Toast.makeText(context, "亲, Wifi已经关闭, 不用再关了", Toast.LENGTH_SHORT).sh
        ow();
48.     } else if (mWifiManager.getWifiState() == 0) {
49.         Toast.makeText(context, "亲, Wifi正在关闭, 不用再关了", Toast.LENGTH_S
        how(T).sh
        ow();
50.     } else {
51.         Toast.makeText(context, "请重新关闭", Toast.LENGTH_SHORT).show();
52.     }
53. }
54.
55. // 检查当前WIFI状态
56. public void checkState(Context context) {
57.     if (mWifiManager.getWifiState() == 0) {
58.         Toast.makeText(context, "Wifi正在关闭", Toast.LENGTH_SHORT).show();
59.     } else if (mWifiManager.getWifiState() == 1) {
60.         Toast.makeText(context, "Wifi已经关闭", Toast.LENGTH_SHORT).show();
61.     } else if (mWifiManager.getWifiState() == 2) {
62.         Toast.makeText(context, "Wifi正在开启", Toast.LENGTH_SHORT).show();
63.     } else if (mWifiManager.getWifiState() == 3) {
64.         Toast.makeText(context, "Wifi已经开启", Toast.LENGTH_SHORT).show();
65.     } else {
66.         Toast.makeText(context, "没有获取到Wifi状态", Toast.LENGTH_SHORT).show();
67.     }
68. }
69.
70. // 锁定WifiLock
71. public void acquireWifiLock() {
72.     mWifiLock.acquire();
73. }

```



关闭



少儿编程





Android 蓝牙开发 (七) hf...

TTcccCarrie : 蓝牙通话时选择蓝牙, 会调到  
switchInCallAudio()

Android 蓝牙开发 (七) hf...

TTcccCarrie : 蓝牙通话时进行的音频连接

Android 蓝牙开发 (六) hf...

TTcccCarrie : hfp连接过程已经分析完了,  
而断开连接到过程

## 文章搜索



## 装修公司



## 联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服    💬 客服论坛

```
74.
75. // 解锁WifiLock
76. public void releaseWifiLock() {
77.     // 判断时候锁定
78.     if (mWifiLock.isHeld()) {
79.         mWifiLock.acquire();
80.     }
81. }
82.
83. // 创建一个WifiLock
84. public void creatWifiLock() {
85.     mWifiLock = mWifiManager.createWifiLock("Test");
86. }
87.
88. // 得到配置好的网络
89. public List<WifiConfiguration> getConfiguration() {
90.     return mWifiConfiguration;
91. }
92.
93. // 指定配置好的网络进行连接
94. public void connectConfiguration(int index) {
95.     // 索引大于配置好的网络索引返回
96.     if (index > mWifiConfiguration.size()) {
97.         return;
98.     }
99.     // 连接配置好的指定ID的网络
100.    mWifiManager.enableNetwork(mWifiConfigurati
101.                               true);
102. }
103.
104. public void startScan(Context context) {
105.    mWifiManager.startScan();
106.    // 得到扫描结果
107.    mWifiList = mWifiManager.getScanResults();
108.    // 得到配置好的网络连接
109.    mWifiConfiguration = mWifiManager.getConfig
110.    if (mWifiList == null) {
111.        if(mWifiManager.getWifiState()==3){
```



18




关闭



## 少儿编程



关于 招聘 广告服务  百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```

112.         Toast.makeText(context, "当前区域没有无线网络", Toast.LENGTH_SHORT).show
113.     ());
114.         }else if(mWifiManager.getWifiState()==2){
115.             Toast.makeText(context, "WiFi正在开启, 请稍后重新点击扫描", Toast.LENGTH_S
116. HORT).show();
117.         }else{
118.             Toast.makeText(context, "WiFi没有开启, 无法扫描", Toast.LENGTH_SHORT).sho
119. w());
120.         }
121.     }
122. }
123.
124. // 得到网络列表
125. public List<ScanResult> getWifiList() {
126.     return mWifiList;
127. }
128.
129. // 查看扫描结果
130. public StringBuilder lookUpScan() {
131.     StringBuilder stringBuilder = new StringBuilder();
132.     for (int i = 0; i < mWifiList.size(); i++) {
133.         stringBuilder
134.             .append("Index_" + new Integer(i + 1).toString() + ":");
135.         // 将ScanResult信息转换成一个字符串包
136.         // 其中把包括: BSSID、SSID、capabilities、frequency、level
137.         stringBuilder.append((mWifiList.get(i)).
138.             stringBuilder.append("/n");
139.     }
140.     return stringBuilder;
141. }
142.
143. // 得到MAC地址
144. public String getMacAddress() {
145.     return (mWifiInfo == null) ? "NULL" : mWifi
146. }
147.
148. // 得到接入点的BSSID
149. public String getBSSID() {
150.     return (mWifiInfo == null) ? "NULL" : mWifi

```



18



关闭



少儿编程



```
148.     }
149.
150.     // 得到IP地址
151.     public int getIPAddress() {
152.         return (mWifiInfo == null) ? 0 : mWifiInfo.getIpAddress();
153.     }
154.
155.     // 得到连接的ID
156.     public int getNetworkId() {
157.         return (mWifiInfo == null) ? 0 : mWifiInfo.getNetworkId();
158.     }
159.
160.     // 得到WifiInfo的所有信息包
161.     public String getWifiInfo() {
162.         return (mWifiInfo == null) ? "NULL" : mWifiInfo.toString();
163.     }
164.
165.     // 添加一个网络并连接
166.     public void addNetwork(WifiConfiguration wcg) {
167.         int wcgID = mWifiManager.addNetwork(wcg);
168.         boolean b = mWifiManager.enableNetwork(wcgID, true);
169.         System.out.println("a--" + wcgID);
170.         System.out.println("b--" + b);
171.     }
172.
173.     // 断开指定ID的网络
174.     public void disconnectWifi(int netId) {
175.         mWifiManager.disableNetwork(netId);
176.         mWifiManager.disconnect();
177.     }
178.     public void removeWifi(int netId) {
179.         disconnectWifi(netId);
180.         mWifiManager.removeNetwork(netId);
181.     }
182.
183.     //然后是一个实际应用方法，只验证过没有密码的情况：
184.
185.     public WifiConfiguration CreateWifiInfo(String
```

e)



18



关闭



少儿编程



```

186.     {
187.         WifiConfiguration config = new WifiConfiguration();
188.         config.allowedAuthAlgorithms.clear();
189.         config.allowedGroupCiphers.clear();
190.         config.allowedKeyManagement.clear();
191.         config.allowedPairwiseCiphers.clear();
192.         config.allowedProtocols.clear();
193.         config.SSID = "\"" + SSID + "\"";
194.
195.         WifiConfiguration tempConfig = this.IsExsits(SSID);
196.         if(tempConfig != null) {
197.             mWifiManager.removeNetwork(tempConfig.networkId);
198.         }
199.
200.         if(Type == 1) //WIFICIPHER_NOPASS
201.         {
202.             config.wepKeys[0] = "";
203.             config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
204.             config.wepTxKeyIndex = 0;
205.         }
206.         if(Type == 2) //WIFICIPHER_WEP
207.         {
208.             config.hiddenSSID = true;
209.             config.wepKeys[0] = "\"" + Password + "\"";
210.             config.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.SHARED);
211.         }
212.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
213.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
214.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.WEP40);
215.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.WEP128);
216.         config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WEP40);
217.         config.wepTxKeyIndex = 0;
218.         if(Type == 3) //WIFICIPHER_WPA
219.         {
220.             config.preSharedKey = "\"" + Password + "\"";
221.             config.hiddenSSID = true;

```



18



关闭



少儿编程





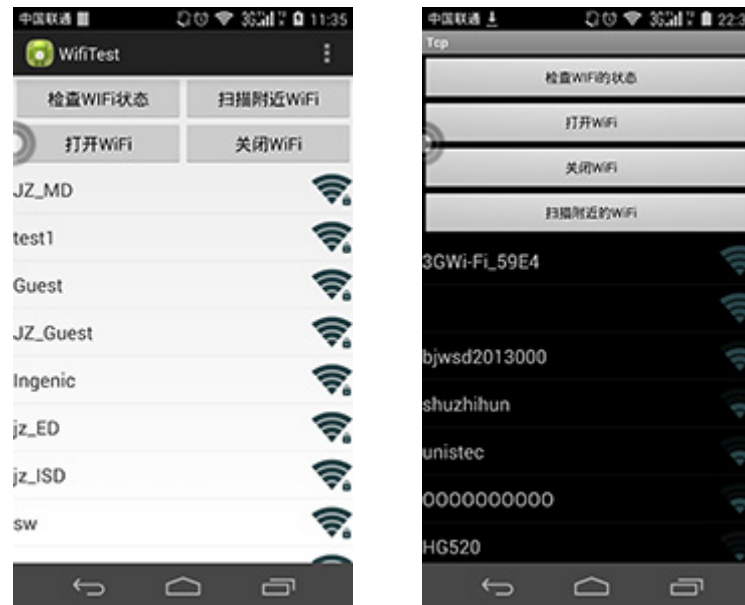
```
222.         config.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN
223.     );
224.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP
225.     );
226.         config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK
227.     );
228.         config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP
229.     );
230.         //config.allowedProtocols.set(WifiConfiguration.Protocol.WPA);
231.         config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMF
232.     );
233.         config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMF
234.     );
235.         config.status = WifiConfiguration.Status.ENABLED;
236.     }
237.     return config;
238. }
239.
240. private WifiConfiguration IsExsits(String SSID)
241. {
242.     List<WifiConfiguration> existingConfigs = mWifiManager.getConfiguredNetworks
243. ();
244.     for (WifiConfiguration existingConfig : existingConfigs)
245.     {
246.         if (existingConfig.SSID.equals("\""+SSID+"\""))
247.         {
248.             return existingConfig;
249.         }
250.     }
251.     return null;
252. }
```

如图所示：



少儿编程





18



## 俩bug

这些代码看起来没有什么问题，但是通过不同环境的测试，发现了一些问题。

1 测试到的wifi热点，有的ssid为“”，也就是ssid != null,获取不到said。

2 wifi列表中有许多同名的wifi热点，也就是扫描的结果中有重合部分

第一个问题ssid为“”，这个看设置中并没有多余wifi，但这个热点点的，应该是该热点隐藏了，所以获取不到。这也就是手机设置中为什么

第二个问题，当附近wifi热点比较少时不会出现此问题，当附近wifi网名的热点进行删除，但是如果真有两个ssid名相同的wifi，那就可以通也相同就没办法了，系统设置里面也不显示同名的。

修改上面的方法 startScan ( )。

关闭



### 少儿编程



[java]

```

1. public void startScan(Context context) {
2.     mWifiManager.startScan();
3.     //得到扫描结果
4.     List<ScanResult> results = mWifiManager.getScanResults();
5.     // 得到配置好的网络连接
6.     mWifiConfiguration = mWifiManager.getConfiguredNetworks();
7.     if (results == null) {
8.         if(mWifiManager.getWifiState()==3){
9.             Toast.makeText(context, "当前区域没有无线网络", Toast.LENGTH_SHORT).show();
10.        }else if(mWifiManager.getWifiState()==2){
11.            Toast.makeText(context, "wifi正在开启, 请稍后扫描", Toast.LENGTH_SHORT).show();
12.        }else{Toast.makeText(context, "WiFi没有开启", Toast.LENGTH_SHORT).show();
13.        }
14.    } else {
15.        mWifiList = new ArrayList();
16.        for(ScanResult result : results){
17.            if (result.SSID == null || result.SSID.length() == 0 || result.capabilities.contains("[IBSS]")) {
18.                continue;
19.            }
20.            boolean found = false;
21.            for(ScanResult item:mWifiList){
22.                if(item.SSID.equals(result.SSID) || item.capabilities.equals(result.capabilities)){
23.                    found = true;break;
24.                }
25.            }
26.            if(!found){
27.                mWifiList.add(result);
28.            }
29.        }
30.    }
31. }

```



18



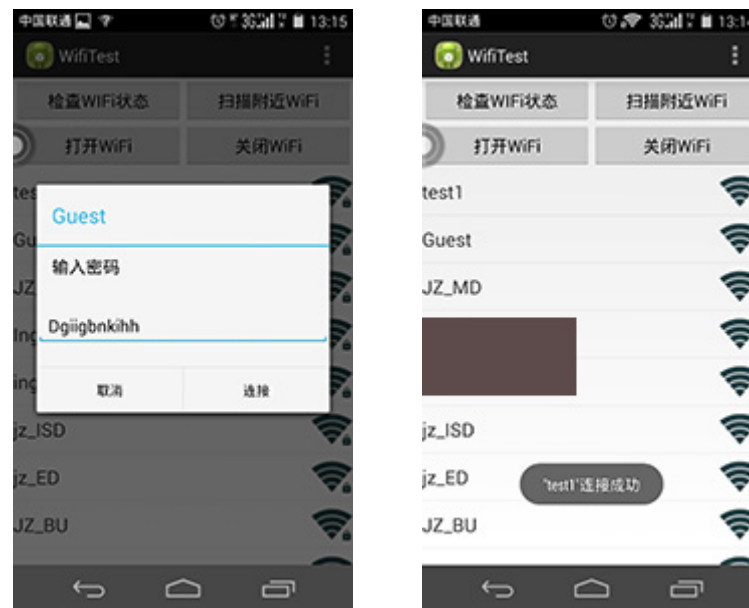
关闭



少儿编程



这样就可以避免出现上面的两种情况了。



18



代码

[java]

```
1. public class MainActivity extends Activity implements
2.     public static final String TAG = "MainActivity"
3.     private Button check_wifi, open_wifi, close_wifi,
4.     private ListView mListview;
5.     protected WifiAdmin mWifiAdmin;
6.     private List<ScanResult> mWifiList;
7.     public int level;
8.     protected String ssid;
9.
10.    protected void onCreate(Bundle savedInstanceState) {
11.        super.onCreate(savedInstanceState);
12.        setContentView(R.layout.activity_main);
```



少儿编程



```

13.         mWifiAdmin = new WifiAdmin(MainActivity.this);
14.         initView();
15.         IntentFilter filter = new IntentFilter(
16.             WifiManager.NETWORK_STATE_CHANGED_ACTION);
17.         //="android.net.wifi.STATE_CHANGE" 监听wifi状态的变化
18.         registerReceiver(mReceiver, filter);
19.         mListview.setOnItemClickListener(new OnItemClickListener() {
20.             @Override
21.             public void onItemClick(AdapterView<?> parent, View view,
22.                 int position, long id) {
23.                 AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
24.                 ssid=mWifiList.get(position).SSID;
25.                 alert.setTitle(ssid);
26.                 alert.setMessage("输入密码");
27.                 final EditText et_password=new EditText(MainActivity.this);
28.                 final SharedPreferences preferences=getSharedPreferences("wifi_password", Context.MODE_PRIVATE);
29.                 et_password.setText(preferences.getString(ssid, ""));
30.                 alert.setView(et_password);
31.                 //alert.setView(view1);
32.                 alert.setPositiveButton("连接", new DialogInterface.OnClickListener()
33.                 {
34.                     @Override
35.                     public void onClick(DialogInterface dialog, int which) {
36.                         String pw = et_password.getText().toString();
37.                         if(null == pw || pw.length() < 8) {
38.                             Toast.makeText(MainActivity.this, "密码长度不能小于8位", Toast.LENGTH_SHORT).show();
39.                             return;
40.                         }
41.                         Editor editor=preferences.edit();
42.                         editor.putString(ssid, pw);
43.                         editor.commit();
44.                         mWifiAdmin.addNetwork(mWifiAdmin.newNetworkBuilder().setSsid(ssid).setPassword(pw).setNetworkId(mWifiAdmin.getNetworkId().toString(), 3));
45.                     }
46.                 });

```



18

asswo

关闭



少儿编程





```

46.         alert.setNegativeButton("取消", new DialogInterface.OnClickListener()
47.         {
48.             @Override
49.             public void onClick(DialogInterface dialog, int which) {
50.                 //
51.                 //mWifiAdmin.removeWifi(mWifiAdmin.getNetworkId());
52.             }
53.         });
54.         alert.create();
55.         alert.show();
56.     }
57. });
58. }
59.
60. /*
61.  * 控件初始化
62.  * */
63. private void initView() {
64.     check_wifi=(Button) findViewById(R.id.check_wifi);
65.     open_wifi=(Button) findViewById(R.id.open_wifi);
66.     close_wifi=(Button) findViewById(R.id.close_wifi);
67.     scan_wifi=(Button) findViewById(R.id.scan_wifi);
68.     mListview=(ListView) findViewById(R.id.wifi_list);
69.     check_wifi.setOnClickListener(MainActivity.this);
70.     open_wifi.setOnClickListener(MainActivity.this);
71.     close_wifi.setOnClickListener(MainActivity.this);
72.     scan_wifi.setOnClickListener(MainActivity.this);
73. }
74.
75. @Override
76. public void onClick(View v) {
77.     switch (v.getId()) {
78.         case R.id.check_wifi:
79.             mWifiAdmin.checkState(MainActivity.this);
80.             break;
81.         case R.id.open_wifi:
82.             mWifiAdmin.openWifi(MainActivity.this);
83.             break;

```



18



关闭



少儿编程



```

84.         case R.id.close_wifi:
85.             mWifiAdmin.closeWifi(MainActivity.this);
86.             break;
87.         case R.id.scan_wifi:
88.             mWifiAdmin.startScan(MainActivity.this);
89.             mWifiList=mWifiAdmin.getWifiList();
90.             if(mWifiList!=null){
91.                 mListview.setAdapter(new MyAdapter(this,mWifiList));
92.                 new Utility().setListViewHeightBasedOnChildren(mListview);
93.             }
94.             break;
95.         default:
96.             break;
97.     }
98. }
99.
100. public class MyAdapter extends BaseAdapter{
101.     LayoutInflater inflater;
102.     List<ScanResult> list;
103.     public MyAdapter(Context context, List<ScanResult> list){
104.         this.inflater=LayoutInflater.from(context);
105.         this.list=list;
106.     }
107.     @Override
108.     public int getCount() {
109.         return list.size();
110.     }
111.     @Override
112.     public Object getItem(int position) {
113.         return position;
114.     }
115.     @Override
116.     public long getItemId(int position) {
117.         return position;
118.     }
119.     @SuppressWarnings("ViewHolder", "InflateParam")
120.     @Override
121.     public View getView(int position, View convertView,
122.         ViewGroup parent) {
123.         View view=null;

```



18



关闭



少儿编程



```

123.         view=inflater.inflate(R.layout.wifi_listitem, null);
124.         ScanResult scanResult = list.get(position);
125.         TextView wifi_ssid=(TextView) view.findViewById(R.id.ssid);
126.         ImageView wifi_level=(ImageView) view.findViewById(R.id.wifi_level);
127.         wifi_ssid.setText(scanResult.SSID);
128.         Log.i(TAG, "scanResult.SSID="+scanResult);
129.         level=WifiManager.calculateSignalLevel(scanResult.level,5);
130.         if(scanResult.capabilities.contains("WEP")||scanResult.capabilities.contains("PSK")||
131.             scanResult.capabilities.contains("EAP")){
132.             wifi_level.setImageResource(R.drawable.wifi_signal_lock);
133.         }else{
134.             wifi_level.setImageResource(R.drawable.wifi_signal_open);
135.         }
136.         wifi_level.setImageLevel(level);
137.         //判断信号强度，显示对应的指示图标
138.         return view;
139.     }
140. }
141.
142. /*设置listview的高度*/
143. public class Utility {
144.     public void setListViewHeightBasedOnChildren(ListView listView) {
145.         ListAdapter listAdapter = listView.getAdapter();
146.         if (listAdapter == null) {
147.             return;
148.         }
149.         int totalHeight = 0;
150.         for (int i = 0; i < listAdapter.getCount(); i++) {
151.             View listItem = listAdapter.getView(i, null, listView);
152.             listItem.measure(0, 0);
153.             totalHeight += listItem.getMeasuredHeight();
154.         }
155.         ViewGroup.LayoutParams params = listView.getLayoutParams();
156.         params.height = totalHeight + (listView.getPaddingTop() + listView.getPaddingBottom());
157.         listView.setLayoutParams(params);
158.     }
159. }

```



18



关闭



少儿编程



```
160. //监听wifi状态
161. private BroadcastReceiver mReceiver = new BroadcastReceiver (){
162.     @Override
163.     public void onReceive(Context context, Intent intent) {
164.         ConnectivityManager manager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
165.
166.         NetworkInfo wifiInfo = manager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
167.         if(wifiInfo.isConnected()){
168.             WifiManager wifiManager = (WifiManager) context
169.                 .getSystemService(Context.WIFI_SERVICE);
170.             String wifiSSID = wifiManager.getConnectionInfo()
171.                 .getSSID();
172.             Toast.makeText(context, wifiSSID+"连接成功", 1).show();
173.         }
174.     }
175. };
176. };
177. }
```



18



## ScanResult类

```
public class
ScanResult
extends Object
implements Parcelable

java.lang.Object
↳ android.net.wifi.ScanResult
```


这个类主要是通过Wifi硬件的扫描来获取一些周边的wifi热点（access





## 少儿编程



返回类型	域名	解释
public String	BSSID	接入点的地址
public String	SSID	网络的名字
public String	capabilities	网络性能，包括接入点支持的认证、密钥管理、加密机制等
public int	frequency	以MHz为单位的接入频率
public int	level	以dBm为单位的信号强度。

  
18





打印信息如下所示：

```
SSID = shuzhihun BSSID=44:7b:c4:00:72:75
capabilities=[WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] frequency=2412
level=-64
```

WifiConfiguration类

通过该类获取一个wifi网络的网络配置，包括安全配  
示：

子类	解释
WifiConfiguration.AuthAlgorithm	获取IEEE
WifiConfiguration.GroupCipher	获取组密



少儿编程





WifiConfiguration.KeyMgmt	获取密码管理体制
WifiConfiguration.PairwiseCipher	获取WPA方式的成对密钥
WifiConfiguration.Protocol	获取加密协议
WifiConfiguration.Status	获取当前网络状态



18



该类内容较多，不一一例举，需要用到的时候可以查Android SDK。

WifiInfo类

```
public class
WifiInfo
extends Object
implements Parcelable

java.lang.Object
↳ android.net.wifi.WifiInfo
```

该类可以获得已经建立的或处于活动状态的wifi网络的:

关闭



少儿编程



Public Methods		
String	getBSSID ()	Return the basic service set identifier (BSSID) of the current access point.
static NetworkInfo.DetailedState	getDetailedStateOf (SupplicantState suppState)	Map a supplicant state into a fine-grained network connectivity state.
boolean	getHiddenSSID ()	
int	getIpAddress ()	
int	getLinkSpeed ()	Returns the current link speed in LINK_SPEED_UNITS.
String	getMacAddress ()	
int	getNetworkId ()	Each configured network has a unique small integer ID, used to identify the network when performing operations on the supplicant.
int	getRssi ()	Returns the received signal strength indicator of the current 802.11 network.
String	getSSID ()	Returns the service set identifier (SSID) of the current 802.11 network.
SupplicantState	getSupplicantState ()	Return the detailed state of the supplicant's negotiation with an access point, in the form of a SupplicantState.



关闭



少儿编程



代码：

```
[java]
1. mWifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
2. mWifiInfo = mWifiManager.getConnectionInfo();
3. if(null != mWifiInfo && null != mWifiInfo.getSSID()){
4.     String info = "getSSID()="+mWifiInfo.getSSID()+"\n"
5.         +"getBSSID()="+mWifiInfo.getBSSID()+"\n"
6.         +"getHiddenSSID()="+mWifiInfo.getHiddenSSID()+"\n"
7.         +"getLinkSpeed()="+mWifiInfo.getLinkSpeed()+"\n"
8.         +"getMacAddress()="+mWifiInfo.getMacAddress()+"\n"
9.         +"getNetworkId()="+mWifiInfo.getNetworkId()+"\n"
10.        +"getRssi()="+mWifiInfo.getRssi()+"\n"
11.        +"getSupplicantState()="+mWifiInfo.getSupplicantState()+"\n"
12.        +"getDetailedStateOf()="+mWifiInfo.getDetailedStateOf(mWifiInfo.getSupplicantState());
13.        mTVWifiInfo.setText(info);
14. }else {
15.     mTVWifiInfo.setText("没有连接到wifi");
16. }
```



18



## WifiManager类

该类用于管理Wifi连接，定义了许多常量和方法，这里就不一一说了。

常用方法。



少儿编程



Public Methods	
int	<b>addNetwork</b> (WifiConfiguration config) Add a new network description to the set of configured networks.
static int	<b>calculateSignalLevel</b> (int rssi, int numLevels) Calculates the level of the signal.
static int	<b>compareSignalLevel</b> (int rssiA, int rssiB) Compares two signal strengths.
WifiManager.MulticastLock	<b>createMulticastLock</b> (String tag) Create a new MulticastLock
WifiManager.WifiLock	<b>createWifiLock</b> (int lockType, String tag) Creates a new WifiLock.
WifiManager.WifiLock	<b>createWifiLock</b> (String tag) Creates a new WifiLock.
boolean	<b>disableNetwork</b> (int netId) Disable a configured network.
boolean	<b>disconnect</b> () Disassociate from the currently active access point.
boolean	<b>enableNetwork</b> (int netId, boolean disableOthers) Allow a previously configured network to be associated with.
List<WifiConfiguration>	<b>getConfiguredNetworks</b> () Return a list of all the networks configured.
WifiInfo	<b>getConnectionInfo</b> () Return dynamic information about the current connection.
DhcpInfo	<b>getDhcpInfo</b> () Return the DHCP-assigned addresses for the current connection.
List<ScanResult>	<b>getScanResults</b> () Return the results of the latest access point scan.
int	<b>getWifiState</b> () Gets the Wi-Fi enabled state.
boolean	<b>isWifiEnabled</b> () Return whether Wi-Fi is enabled or disabled.



18



关闭



少儿编程



boolean	<code>isWifiEnabled()</code> Return whether Wi-Fi is enabled or disabled.
boolean	<code>pingSupplicant()</code> Check that the supplicant daemon is responding to requests.
boolean	<code>reassociate()</code> Reconnect to the currently active access point, even if we are already connected.
boolean	<code>reconnect()</code> Reconnect to the currently active access point, if we are currently disconnected.
boolean	<code>removeNetwork(int netId)</code> Remove the specified network from the list of configured networks.
boolean	<code>saveConfiguration()</code> Tell the supplicant to persist the current list of configured networks.
boolean	<code>setWifiEnabled(boolean enabled)</code> Enable or disable Wi-Fi.
boolean	<code>startScan()</code> Request a scan for access points.
int	<code>updateNetwork(WifiConfiguration config)</code> Update the network description of an existing configured network.



18



需要指出的是getWifiState()方法是反映wifi的状态，有整型常量表示，

关闭

int	<code>WIFI_STATE_DISABLED</code>	1
int	<code>WIFI_STATE_DISABLING</code>	0
int	<code>WIFI_STATE_ENABLED</code>	3
int	<code>WIFI_STATE_ENABLING</code>	2
int	<code>WIFI_STATE_UNKNOWN</code>	4



少儿编程



demo下载地址



- 上一篇 Objective-C 11 属性property
- 下一篇 Android WiFi开发（二）Wifi热点



局域网屏幕监控



北大青鸟



初音未来歌曲



公众号怎么推广



3D 扫描仪



计算



您还没有登录,请[登录](#)或[注册](#)

查看评论



枫叶丶

23楼 2018-04-10 11:00 发表

关闭

博主你好，我把你的代码写了一遍，请问一下有这方面的书籍可推荐吗？不然光这样写写再多也没有用，还需要了解理论的上的，思路很需要，可以加一个好友QQ:772165619



zpengyong

Re: 20

回复枫叶丶：这些功能，看api文档就可以了，加上实践。

[查看更多评论](#)



少儿编程



## Android WiFi模块功能开发

Android wifi模块开发



free092875

2016年10月11日 16:06

4710

### Android WiFi开发教程（二）——WiFi的搜索和连接

在上一篇中我们介绍了WiFi热点的创建和关闭，如果你  
还没阅读过，建议先阅读上一篇文章Android WiFi开发教

程（一）——WiFi热点的创建与关闭。本章节主要继续介绍WiFi的搜索和连接。WiF...



a153358867

2016年09月07日 15:2

5231



18



### 论一个普通设计师与高级设计师的区别

你以为只是PS水平的差距么？天真！



### Android/安卓开发之WIFI的基本应用



qq\_3600

关闭

本文没有涉及到连接WIFI之后进行通讯，若有这方面的想法（例如两  
请关注后续文章一起讨论。其实关于WIFI的开发的文

### Android WiFi开发教程（一）——WiFi热点的创

相对于BlueTooth，WiFi是当今使用最广的一种无线网络  
传输技术，几乎所有智能手机、平板电脑和笔记本电脑都

支持Wi-Fi上网。因此，掌握基本的WiFi开发技术是非常必要的。本教



a153358



### 少儿编程



## android之wifi开发（一）



wangkuifeng0118

2012年03月10日 14:54

106593

WIFI就是一种无线联网技术，常见的是使用无线路由器。那么在这个无线路由器的信号覆盖的范围内都可以采用WIFI连接的方式进行联网。如果无线路由器连接了一个ADSL线路或其他的联网线路，则又被称为“热点...

## 码农不会英语怎么行？英语文档都看不懂！

软件工程出身的英语老师，教你用数学公式读懂天下英文→



## Android开发--WIFI实现



a123473915

2017年01月21日 11:54

954

<http://blog.csdn.net/jdsjlzx/article/details/40740543> wifi的基本结构 在Android的官方文档中定义了如下五种状态：

...

## Android 蓝牙通信及WiFi开发



zokv

2017年02月16日 11:21

1120

在我们正常的Android蓝牙功能开发步骤中，一般要经过系统权限和蓝牙数据通信这几个过程。在Android 4.3系统之后，我们可以使用蓝牙

关闭

## Android中WIFI开发总结



ye

WIFI就是一种无线联网技术，常见的是使用无线路由器。那么在这个用WIFI连接的方式进行联网。如果无线路由器连接了一个ADSL线路或



### 少儿编程



## Android WiFi/WiFi热点开发总结

首先看一下WiFi的自我介绍：Wi-Fi是一种允许电子设备连接到一个无线局域网（WLAN）的技术，通常使用2.4 G UHF或5G SHF ISM 射频频段。连接到无线局域网通常是有密码保护的；但也可...

## Android开发--WIFI实现



jdsjlzx 2014年11月03日 14:29 18040

实现目标：搜索WIFI，手动输入密码并保存，连接WIFI。第二次连接该WIFI信号不需要输入密码 首先在AndroidManifest.XML中开启响应的权限 ...



18

## android网络开发之WIFI篇



aiyuan1996 2016年10月15日 21:2 2672

WIFI层次结构 wifi系统的上层接口包括数据部分和控制部分，数据部分通常是一个和以太网类似的设备，控制部分用于实现接入点操作和安全验证处理 在软件层,wifi系统包括linux内...



## 公众号怎么推广

微商怎么做推广新闻推广公众号推广怎么做

百度广告

关闭

## Android WiFi开发（二）Wifi热点



VNanyesheshou

接着上一篇wifi的扫描连接等，这一篇主要说一下手机开启Wifi热点。  
片： 经测试开启wifi热点（无秘密，wpa安全类型，wpa2安全类型）都

## Android WIFI开发、扫描、连接和密码验证



少儿编程



转载请标明出处： <http://blog.csdn.net/u011974987/article/details/50551580>； 本文出自:【Xiho的博客】 网上关于如何用代码设...

## Android Wifi框架流程分析



King1425 2017年03月31日 19:17 2265

//在 SystemServer 启动的时候,启动WifiService调用关系如下: public static void main(String[] args) { new Sys...



18

## Android开发--实现输入密码连接WIFI

2014年04月06日 23:18 1.42MB

下载



RAR

## 5分钟完成加壳，防止代码反编译

Virbox Protector加壳工具，堪比VMP安全强度的加壳工具



关闭

## android wifi开发（一）



u0143

WIFI就是一种无线联网技术，常见的是使用无线路由器。那么在这个...  
用WIFI连接的方式进行联网。如果无线路由器连接了一个ADSL线路可




少儿编程

## Android 4.1 Netd详细分析（一）概述与应用实例





一.概述 所谓 Netd 就是Network Daemon 的缩写，表示Net  zhgeliang 2017年12月20日 14:44 103  
work守护进程，类似的命名还有很多，例如 Vold（Volumn Deamon），Rild（Radio...

## android底层开发进阶（1）-默认设置项的修改

android系统默认设置在安卓源码中，要对安卓的一些默认  u013983194 2015年11月04日 15:32 3370  
属性配置进行修改的话（比如：修改背光默认，修改默认  
锁屏开关，休眠时间等等），我们可以非常容易地找到一个xml档·其xml档在/frameworkor...



18



## Android WIFI应用简单开发实例

 dlijun 2016年09月06日 18:27 1600

在网上参阅相关资料后，简单修改提炼，完成自己想要的简单开发实例，记之以备查阅。 主要功能：扫描wifi热点，锁定wifi连接，监听wifi打开/关闭及网络连接/断开...



n on wif

## android wifi开发

 guang\_liang\_ 2017年02月15日 22:37 516

android wifi开发

关闭



少儿编程

