第 8 章　缓存清理

第 8 章　缓存清理模块

- 了解缓存清理模块功能
- 掌握如何获取程序的缓存信息
- 掌握如何清理程序的缓存

随着手机的使用时间增长手机中的缓存信息也就越多，这些缓存信息会导致手机运行速度变慢、变卡，而且还会大大占用内存空间。因此，我们经常会定时清理手机缓存，以保持手机状态良好，本章将针对缓存清理模块进行详细讲解。

8.1 模块概述

缓存清理模块主要用于清理所有程序缓存，当扫描完所有程序后出现缓存时，可以点击一键清理，此时会跳转到清理界面逐渐清理，清理完成后会显示成功清理了多少缓存，该模块界面效果如图 8-1 所示。

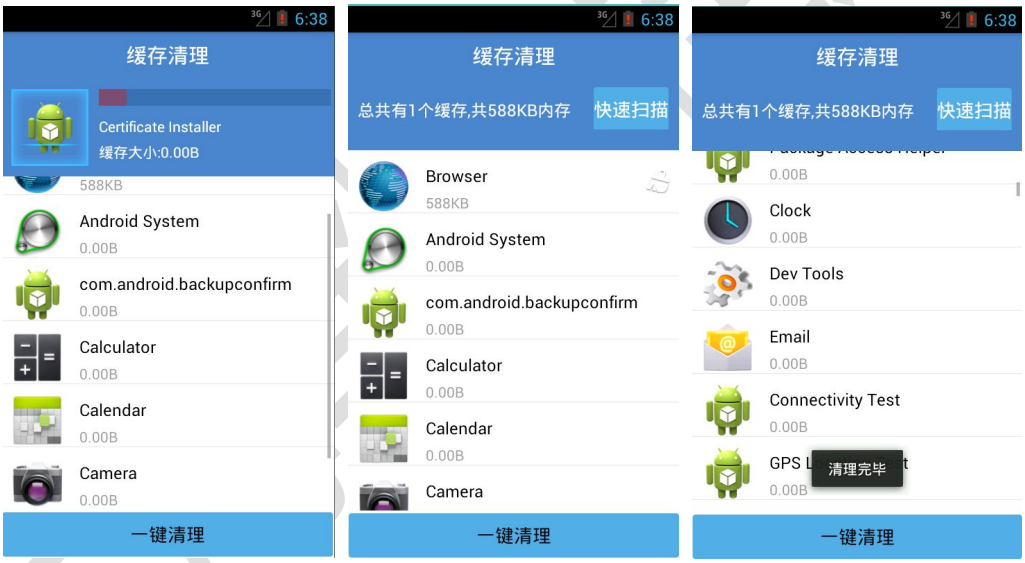

图 8-1 缓存清理界面

8.2 缓存清理之应用信息展示

手机中的大部分程序都有缓存信息，这些缓存信息是通过 AIDL 接口调用系统底层方法获取的，本节将针对扫描缓存功能进行详细讲解。

8.2.1 缓存清理界面 UI

扫描缓存界面，上面是使用相对布局进行多个控件的包裹，中间是 ListView 控件，用于展示扫描完的

程序，最下方是一键清理按钮，扫描缓存的图形化界面如图 8-2 所示。

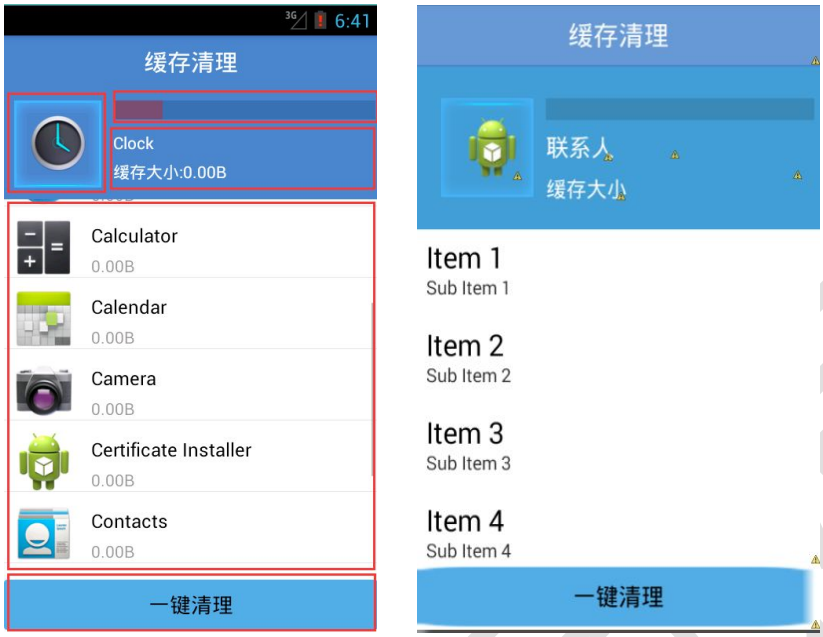

图 8-2 扫描缓存界面

图 8-2 扫描缓界面对应的布局文件如【文件 8-1】所示。

【文件 8-1】 activity_cache.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        style="@style/TitleBarTextView"
        android:gravity="center"
        android:text="缓存清理" />
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="120dp"
        android:background="#429ed6" >
        <RelativeLayout
            android:id="@+id/rl_content"
            android:layout_width="match_parent"
            android:layout_height="120dp"
            android:background="#429ed6" >
            <RelativeLayout
                android:id="@+id/rl_scan_bg"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:layout_marginLeft="20dp"
                android:background="@drawable/scan_bg" >
```

```
26.             <ImageView
27.                 android:id="@+id/iv_icon"
28.                 android:layout_width="50dp"
29.                 android:layout_height="50dp"
30.                 android:layout_centerVertical="true"
31.                 android:layout_marginLeft="15dp"
32.                 android:src="@drawable/ic_launcher" />
33.             <ImageView
34.                 android:id="@+id/iv_scan_line"
35.                 android:layout_width="wrap_content"
36.                 android:layout_height="wrap_content"
37.                 android:layout_alignTop="@id/iv_icon"
38.                 android:layout_marginLeft="6dp"
39.                 android:layout_marginTop="2dp"
40.                 android:src="@drawable/scan_line" />
41.         </RelativeLayout>
42.         <TextView
43.             android:id="@+id/tv_app_name"
44.             android:layout_width="wrap_content"
45.             android:layout_height="wrap_content"
46.             android:layout_centerVertical="true"
47.             android:layout_marginLeft="10dp"
48.             android:layout_marginTop="5dp"
49.             android:layout_toRightOf="@id/rl_scan_bg"
50.             android:singleLine="true"
51.             android:text="联系人"
52.             android:textColor="#fff"
53.             android:textSize="18sp" />
54.         <TextView
55.             android:id="@+id/cache_size"
56.             android:layout_width="wrap_content"
57.             android:layout_height="wrap_content"
58.             android:layout_below="@id/tv_app_name"
59.             android:layout_marginLeft="10dp"
60.             android:layout_marginTop="10dp"
61.             android:layout_toRightOf="@id/rl_scan_bg"
62.             android:text="缓存大小"
63.             android:textColor="#fff"
64.             android:textSize="16sp" />
65.         <ProgressBar
66.             android:id="@+id/progressBar"
67.             style="?android:attr/progressBarStyleHorizontal"
68.             android:layout_width="match_parent"
69.             android:layout_height="wrap_content"
```

```
70.              android:layout_alignLeft="@+id/tv_app_name"
71.              android:layout_alignParentRight="true"
72.              android:layout_alignTop="@+id/rl_scan_bg"
73.              android:layout_marginRight="5dp"
74.              android:progressDrawable="@drawable/progress_horizontal" />
75.        </RelativeLayout>
76.    </FrameLayout>
77.    <ListView
78.        android:id="@+id/list_view"
79.        android:layout_width="match_parent"
80.        android:layout_height="match_parent"
81.        android:layout_weight="1" >
82.    </ListView>
83.    <Button
84.        android:id="@+id/bt_clean_cache"
85.        android:layout_width="match_parent"
86.        android:layout_height="wrap_content"
87.        android:background="@drawable/dg_btn_confirm_selector"
88.        android:text="一键清理" />
89. </LinearLayout>
```

上述布局文件中，上方的相对布局包含两个 ImageView 和两个 TextView 控件，还有一个进度条，其中 ImageView 用于展示扫描图标，第一个 TextView 显示正在扫描的程序，第二个 TextView 显示总缓存大小，进度条用于显示当前的扫描进度。中间的 ListView 控件用于显示已扫描缓存的应用列表。下方的线性布局中放置一个 Button 按钮用于点击之后进行清理缓存操作。

其中，progress_horizontal.xml 是用于设置进度条的显示效果，具体代码如【文件 8-2】所示。

【文件 8-2】 res/layout/progress_horizontal.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
3.      <!-- layer-list 表示层级关系 -->
4.      <!-- 背景 -->
5.      <item android:id="@android:id/background">
6.          <shape>
7.              <!-- corners 边角 -->
8.              <!-- <corners android:radius="5dip" /> -->
9.              <!--
10.         梯度渐变色   android:angle="270" 角度 逆时针旋转 270 度
11.             -->
12.             <!-- <gradient -->
13.             <!-- android:startColor="#ff9d9e9d" -->
14.             <!-- android:centerColor="#ff5a5d5a" -->
15.             <!-- android:centerY="0.75" -->
16.             <!-- android:endColor="#ff747674" -->
17.             <!-- android:angle="270" -->
18.             <!-- /> -->
19.             <!-- solid 固定色 -->
```

```
            <solid android:color="#22000000" />
        </shape>
    </item>
    <!-- 副进度条（第二进度条） -->
    <item android:id="@android:id/secondaryProgress">
        <clip>
            <shape>
                <!-- <corners android:radius="5dip" /> -->
                <!-- <gradient -->
                <!-- android:startColor="#80ffd300" -->
                <!-- android:centerColor="#80ffb600" -->
                <!-- android:centerY="0.75" -->
                <!-- android:endColor="#a0ffcb00" -->
                <!-- android:angle="270" -->
                <!-- /> -->
                <solid android:color="#22ff0000" />
            </shape>
        </clip>
    </item>
    <!-- 主进度条 -->
    <item android:id="@android:id/progress">
        <clip>
            <shape>
                <!-- <corners android:radius="5dip" /> -->
                <!-- <gradient -->
                <!-- android:angle="270" -->
                <!-- android:centerColor="#ffffb600" -->
                <!-- android:centerY="0.75" -->
                <!-- android:endColor="#ffffcb00" -->
                <!-- android:startColor="#ffffd300" /> -->
                <solid android:color="#55ff0000" />
            </shape>
        </clip>
    </item>
</layer-list>
```

另外，针对快速扫描按钮和一键清理按钮，我们设置了一样的选择器 dg_btn_confirm_selected.xml，具体代码如【文件 8-3】所示。

【文件 8-3】　res/drawable/dg_btn_confirm_selectedxml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/dg_btn_confirm_select" android:state_pressed=
        "true"></item>
    <item android:drawable="@drawable/dg_btn_confirm_normal"></item>
</selector>
```

8.2.2 缓存清理之扫描动画

观察完整版应用的效果图如图 8-3 所示，发现在扫描应用的时候，左上角的应用上面有一条上下移动的线，这其实是位移动画实现的。

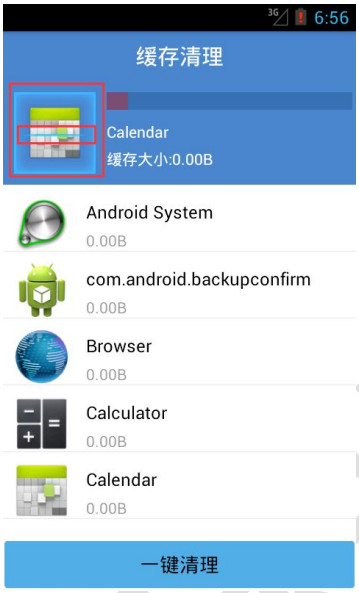

图 8-3　扫描缓存界面

缓存界面 CacheActivity.java 中实现动画的代码如下所示，蓝色字体标注。

```
1.  public class CacheActivity extends Activity {
2.      private ImageView iv_scan_line;
3.      @Override
4.      protected void onCreate(Bundle savedInstanceState) {
5.          super.onCreate(savedInstanceState);
6.          initView();
7.          initData();
8.      }
9.      private class ScanAsyTask extends AsyncTask<Void, Void, Void> {
10.         @Override
11.         protected Void doInBackground(Void... params) {
12.             return null;
13.         }
14.         @Override
15.         protected void onPreExecute() {
16.             super.onPreExecute();
17.             // 第一个参数：相对父亲
18.             TranslateAnimation translateAnimation = new TranslateAnimation(
19.                     Animation.RELATIVE_TO_PARENT, 0,
20.                     Animation.RELATIVE_TO_PARENT, 0,
21.                     Animation.RELATIVE_TO_PARENT, 0f,
22.                     Animation.RELATIVE_TO_PARENT, 0.8f);
23.             translateAnimation.setDuration(800);
```

```
24.             translateAnimation.setRepeatCount(Animation.INFINITE);
25.             translateAnimation.setRepeatMode(Animation.REVERSE);
26.             iv_scan_line.startAnimation(translateAnimation);
27.         }
28.         @Override
29.         protected void onPostExecute(Void result) {
30.             super.onPostExecute(result);
31.         }
32.         @Override
33.         protected void onProgressUpdate(Void... values) {
34.             super.onProgressUpdate(values);
35.         }
36.     }
37.     private void initData() {
38.         ScanAsyTask task = new ScanAsyTask();
39.         task.execute();
40.     }
41.     private void initView() {
42.         setContentView(R.layout.activity_cache);
43.         iv_scan_line = (ImageView) findViewById(R.id.iv_scan_line);
44.     }
45. }
```

运行程序，效果图如图 8-4 所示。

图 8-4　缓存清理的扫描动画

8.2.3 缓存清理 Item 布局

扫描缓存界面下方使用了一个 ListView 控件，因此需要为其定义一个 Item 布局，如图 8-5 所示。该布局包含一个 ImageView 控件、两个 TextView 控件和一个 ImageButton。其中 ImageView 控件用于显示程序

图标，第 1 个 TextView 用于显示程序名称，第 2 个 TextView 用于显示有多少缓存，而 ImageButton 用于清理当前应用程序的缓存。具体代码如【文件 8-4】所示。

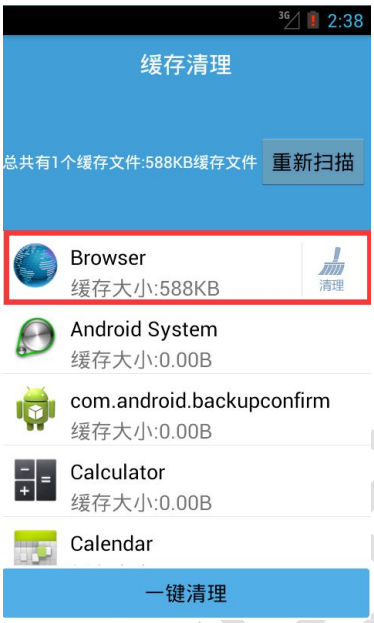

图 8-5　缓存界面 item

【文件 8-4】item_cacheclean_list.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="match_parent"
4.      android:layout_height="wrap_content" >
5.      <ImageView
6.          android:id="@+id/iv_icon"
7.          android:layout_width="40dp"
8.          android:layout_height="40dp"
9.          android:layout_margin="10dp"
10.         android:src="@drawable/ic_launcher" />
11.     <TextView
12.         android:id="@+id/tv_app_name"
13.         android:layout_width="wrap_content"
14.         android:layout_height="wrap_content"
15.         android:layout_marginTop="10dp"
16.         android:layout_toRightOf="@id/iv_icon"
17.         android:text="应用的名字"
18.         android:textColor="#000"
19.         android:textSize="18sp" />
20.     <TextView
21.         android:id="@+id/tv_chache_size"
22.         android:layout_width="wrap_content"
23.         android:layout_height="wrap_content"
24.         android:layout_below="@id/tv_app_name"
25.         android:layout_marginTop="2dp"
```

```
26.         android:layout_toRightOf="@id/iv_icon"
27.         android:text="缓存大小"
28.         android:textColor="#88000000"
29.         android:textSize="18sp" />
30.     <ImageButton
31.         android:id="@+id/ib_list_button_clean"
32.         android:layout_width="wrap_content"
33.         android:layout_height="wrap_content"
34.         android:layout_alignParentRight="true"
35.         android:layout_margin="10dp"
36.         android:src="@drawable/list_button_clean_selector"
37.         android:background="@android:color/transparent" />
38. </RelativeLayout>
```

8.2.4 缓存信息的实体类

由于 ListView 列表展示的是手机中缓存信息，这些信息包括应用包名、缓存大小、应用图标以及该应用名称。因此首先需要根据这些信息定义一个缓存信息的实体类，具体代码如【文件 8-5】所示。

【文件 8-5】CacheInfo.java

```
1.  public class CacheInfo {
2.      public Drawable icon;
3.      public String appName;
4.      public long cacheSize;
5.      public String appPackageName;
6.  }
```

8.2.5 缓存清理之初始化数据

扫描缓存的主要逻辑是获取每个应用程序的缓存大小，然后将所有缓存累加在一起展示在界面上，并展示所有扫描后的应用列表，具体代码如【文件 8-6】所示。

【文件 8-6】 CacheClearListActivity.java

```
1.  public class CacheActivity extends Activity {
2.      private ImageView iv_scan_line;
3.      private PackageManager mPm;
4.      private ScanAsyTask task;
5.      private ListView mListView;
6.      private ImageView iv_icon;
7.      private TextView tv_app_name;
8.      private TextView cache_size;
9.      private CacheAdapter adapter;
10.     // 存放所有的数据
11.     private List<CacheInfo> mDatas;
12.     @Override
```

```
13.     protected void onCreate(Bundle savedInstanceState) {
14.         super.onCreate(savedInstanceState);
15.         mPm = getPackageManager();
16.         initView();
17.         initData();
18.     }
19.     IPackageStatsObserver.Stub mStatsObserver = new IPackageStatsObserver.Stub() {
20.         public void onGetStatsCompleted(PackageStats stats, boolean succeeded) {
21.             try {
22.                 long cacheSize = stats.cacheSize;
23.                 CacheInfo info = new CacheInfo();
24.                 String packageName = stats.packageName;
25.                 PackageInfo packageInfo = mPm.getPackageInfo(packageName, 0);
26.                 Drawable icon = packageInfo.applicationInfo.loadIcon(mPm);
27.                 info.icon = icon;
28.                 String appName = packageInfo.applicationInfo.loadLabel(mPm)
29.                         .toString();
30.                 info.appName = appName;
31.                 info.appPackageName = packageName;
32.                 if (cacheSize > 0) {
33.                     info.cacheSize = cacheSize;
34.                     mDatas.add(0, info);
35.                 } else {
36.                     mDatas.add(info);
37.                 }
38.                 // 更新数据
39.                 task.updateCacheInfo(info);
40.             } catch (Exception e) {
41.                 // TODO Auto-generated catch block
42.                 e.printStackTrace();
43.             }
44.         }
45.     };
46.     private class ScanAsyTask extends AsyncTask<Void, CacheInfo, Void> {
47.         /**
48.          * 是否扫描完成
49.          */
50.         boolean isFinish = false;
51.         @Override
52.         protected Void doInBackground(Void... params) {
53.             List<PackageInfo> packages = mPm.getInstalledPackages(0);
54.             mDatas = new ArrayList<CacheInfo>();
55.             for (PackageInfo packageInfo : packages)
56.                 String packageName = packageInfo.packageName;
57.                 Drawable icon = packageInfo.applicationInfo.loadIcon(mPm);
```

```
58.                 String appName = packageInfo.applicationInfo.loadLabel(mPm)
59.                         .toString();
60.                 try {
61.                     // 获取某个包名对应的应用程序的缓存大小
62.                     Method method = mPm.getClass().getDeclaredMethod(
63.                             "getPackageSizeInfo", String.class,
64.                             IPackageStatsObserver.class);
65.                     method.invoke(mPm, packageName, mStatsObserver);
66.                     SystemClock.sleep(100);
67.                 } catch (Exception e) {
68.                     // TODO Auto-generated catch block
69.                     e.printStackTrace();
70.                 }
71.             }
72.             return null;
73.         }
74.         /**
75.          * 更新
76.          * @param info
77.          */
78.         public void updateCacheInfo(CacheInfo info) {
79.             // 推送进度条
80.             publishProgress(info);
81.         }
82.         @Override
83.         protected void onPreExecute() {
84.             super.onPreExecute();
85.             // 第一个参数：相对父亲
86.             TranslateAnimation translateAnimation = new TranslateAnimation(
87.                     Animation.RELATIVE_TO_PARENT, 0,
88.                     Animation.RELATIVE_TO_PARENT, 0,
89.                     Animation.RELATIVE_TO_PARENT, 0f,
90.                     Animation.RELATIVE_TO_PARENT, 0.8f);
91.             translateAnimation.setDuration(800);
92.             translateAnimation.setRepeatCount(Animation.INFINITE);
93.             translateAnimation.setRepeatMode(Animation.REVERSE);
94.             iv_scan_line.startAnimation(translateAnimation);
95.         }
96.         @Override
97.         protected void onPostExecute(Void result) {
98.             super.onPostExecute(result);
99.             // 滚动到第 0 个位置
100.                mListView.smoothScrollToPosition(0);
101.            }
```

```
102.            protected void onProgressUpdate(CacheInfo... values) {
103.                super.onProgressUpdate(values);
104.                CacheInfo info = values[0];
105.                iv_icon.setImageDrawable(info.icon);
106.                tv_app_name.setText(info.appName);
107.                cache_size.setText(Formatter.formatFileSize(
108.                        getApplicationContext(), info.cacheSize));
109.                if (adapter == null) {
110.                    adapter = new CacheAdapter();
111.                    mListView.setAdapter(adapter);
112.                } else {
113.                    adapter.notifyDataSetChanged();
114.                }
115.                // 自己滚动
116.                mListView.smoothScrollToPosition(adapter.getCount());
117.            }
118.        }
119.        //初始化数据
120.       private void initData() {
121.            task = new ScanAsyTask();
122.            task.execute();
123.        }
124.       //初始化 view 对象
125.        private void initView() {
126.            setContentView(R.layout.activity_cache);
127.            iv_scan_line = (ImageView) findViewById(R.id.iv_scan_line);
128.            mListView = (ListView) findViewById(R.id.list_view);
129.            // 应用的图标
130.            iv_icon = (ImageView) findViewById(R.id.iv_icon);
131.            tv_app_name = (TextView) findViewById(R.id.tv_app_name);
132.            cache_size = (TextView) findViewById(R.id.cache_size);
133.        }
```

- 第 19~45 行实现 android.content.pm.IPackageStatsObserver.Stub 的 AIDL 文件实现类，需要实现该类中的 onGetStatsCompleted()方法，然后通过 pStats.cacheSize()方法就可以获取到缓存信息，应用名称，应用包名，应用图标，最后将扫描过的程序添加到 mDatas 集合中。
- 第 60~71 行用于获取每个程序的缓存大小，由于 getPackageSizeInfo(String packageName,IPackageStatsObserver observer)方法是隐藏的，因此需要通过反射机制获取该方法，该方法参数中的 packageName 表示包名，observer 是一个远程服务的 AIDL 接口。
- 第 86~95 行用于设置缓存清理上边的扫描动画。
- 第 121~135 行进行初始化数据，初始化 view 对象。

需要注意的是，获取缓存时需要使用 IPackageStatsObserver 接口，因此需要创建一个 android.content.pm 包，将 IPackageStatsObserver.aidl 拷贝到工程中，由于该接口还依赖于 PackageStats.aidl 接口，因此也需要将该文件拷贝到 android.content.pm 包中。

在获取程序缓存时，需要在 AndroidManifest.xml 文件中配置相关权限，具体代码如下所示：

```
<uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
```

8.2.6 缓存清理之数据适配器

由于扫描缓存界面使用了 ListView 控件，该控件中需要使用适配器进行数据填充，因此需要定义一个数据适配器，具体代码如下所示。

```
1.  private class CacheAdapter extends BaseAdapter {
2.          @Override
3.          public int getCount() {
4.              // TODO Auto-generated method stub
5.              return mDatas.size();
6.          }
7.          @Override
8.          public Object getItem(int position) {
9.              // TODO Auto-generated method stub
10.             return mDatas.get(position);
11.         }
12.         @Override
13.         public long getItemId(int position) {
14.             // TODO Auto-generated method stub
15.             return position;
16.         }
17.         @Override
18.         public View getView(int position, View convertView, ViewGroup parent) {
19.             if (convertView == null) {
20.                 convertView = View.inflate(getApplicationContext(),
21.                         R.layout.item_cache_activity, null);
22.             }
23.             ImageView iv_icon = (ImageView) convertView
24.                     .findViewById(R.id.iv_icon);
25.             TextView tv_app_name = (TextView) convertView
26.                     .findViewById(R.id.tv_app_name);
27.             TextView tv_chache_size = (TextView) convertView
28.                     .findViewById(R.id.tv_chache_size);
29.             ImageButton ib_list_button_clean = (ImageButton) convertView
30.                     .findViewById(R.id.ib_list_button_clean);
31.             final CacheInfo cacheInfo = mDatas.get(position);
32.             iv_icon.setImageDrawable(cacheInfo.icon);
33.             tv_app_name.setText(cacheInfo.appName);
34.             tv_chache_size.setText("缓存大小:"
35.                     + Formatter.formatFileSize(getApplicationContext(),
36.                             cacheInfo.cacheSize));
37.             // 判断当前的缓存大小是否大于 0 。如果大于 0.那么就显示清理的图标.如果小于 0.就不显示
38.             ib_list_button_clean
39.                     .setVisibility(cacheInfo.cacheSize > 0 ? View.VISIBLE
```

```
                                : View.INVISIBLE);
            return convertView;
        }
    }
```

运行程序，效果图如图 8-6 所示。

图 8-6 缓冲清理主界面

8.2.7 缓存清理之清理应用程序缓存

当扫描到应用程序有缓存时，右边的清理按钮会显示，这里我们需要实现点击清理按钮时清理缓存的功能，在适配器中实现 ListView 中的条目点击事件。具体的代码在适配器的 getView()方法中，如下所示。

```
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            ......
            ib_list_button_clean.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent();
                    // <intent-filter>
                    // <action
                    // android:name="android.settings.APPLICATION_DETAILS_SETTINGS"
                    // />
                    // <category android:name="android.intent.category.DEFAULT"
                    // />
                    // <data android:scheme="package" />
                    // </intent-filter>
            intent.setAction("android.settings.APPLICATION_DETAILS_SETTINGS");
                    intent.addCategory("android.intent.category.DEFAULT");
                    intent.setData(Uri.parse("package:"
                            + cacheInfo.appPackageName));
                    startActivity(intent);
```

```
21.                 }
22.             });
23.             return convertView;
24.         }
```

运行程序，点击浏览器的清理按钮，效果图如图 8-7 所示。

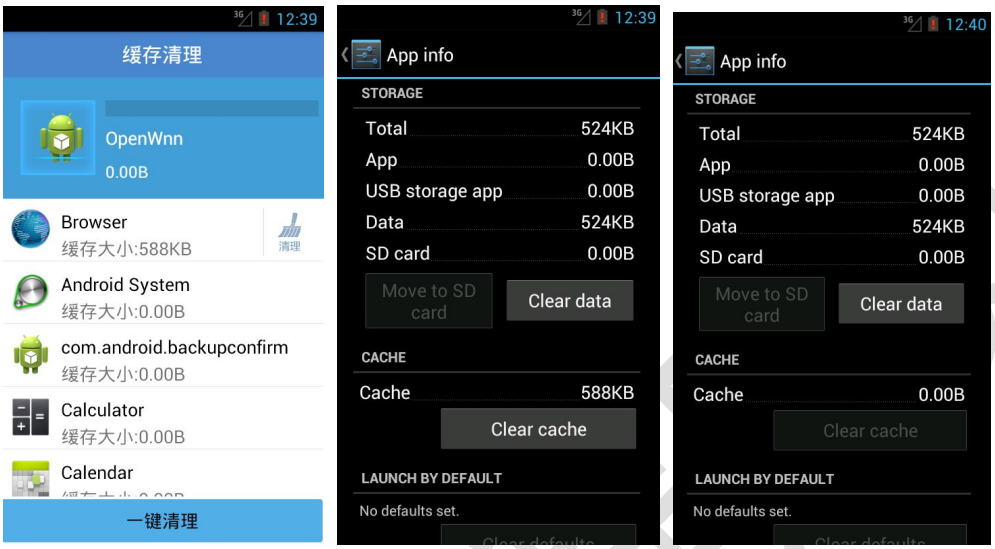

图 8-7 清理应用程序的缓存

8.3 缓存清理之一键清理

缓存清理功能主要是利用 Android 系统漏洞，通过反射隐藏的方法对缓存进行清理的，本节将针对缓存清理功能进行详细讲解。

8.3.1 一键清理

这里要实现一键清理的功能，具体的代码如下所示。

```
1.      private void initView() {
2.          setContentView(R.layout.activity_cache);
3.          ......
4.          // 一键清理
5.          bt_clean_cache = (Button) findViewById(R.id.bt_clean_cache);
6.          bt_clean_cache.setOnClickListener(this);
7.      }
8.      /**
9.       * 一键清理
10.      */
11.     IPackageDataObserver.Stub mDataObserver = new IPackageDataObserver.Stub() {
12.         @Override
13.         public void onRemoveCompleted(String packageName, boolean succeeded)
14.                 throws RemoteException {
15.             Toast.makeText(getApplicationContext(), "清理完毕", 0).show();
```

```
16.         }
17.     };
18.     @Override
19.     public void onClick(View v) {
20.         // TODO Auto-generated method stub
21.         try {
22.             Method method = mPm.getClass().getDeclaredMethod(
23.                     "freeStorageAndNotify", long.class,
24.                     IPackageDataObserver.class);
25.             method.invoke(mPm, Long.MAX_VALUE, mDataObserver);
26.             initData();
27.         } catch (Exception e) {
28.             // TODO Auto-generated catch block
29.             e.printStackTrace();
30.         }
31.     }
```

在清理程序缓存时，需要在 AndroidManifest.xml 文件中配置相关权限，具体代码如下所示：

```
<uses-permission android:name="android.permission.CLEAR_APP_CACHE"/>
```

运行程序，效果图如图 8-8 所示。

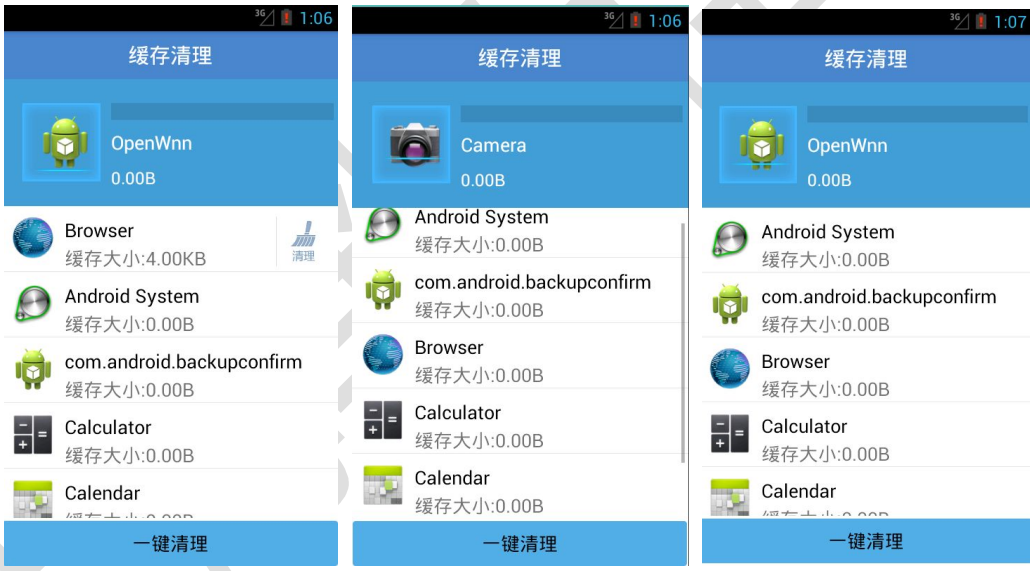

图 8-8 一键清理功能

8.3.2 缓存清理后的完成界面

缓存清理界面其实是由两个界面组成，第一个布局用于展示正在清理缓存界面，第二个布局用于展示缓存清理完成界面，通过控制帧布局的显示与隐藏就可以展示当前在进行哪项操作。默认情况下清理完成的帧布局是隐藏的，这里为了让编程者看到更直观的效果故将其分别进行展示，如图 8-9 所示。

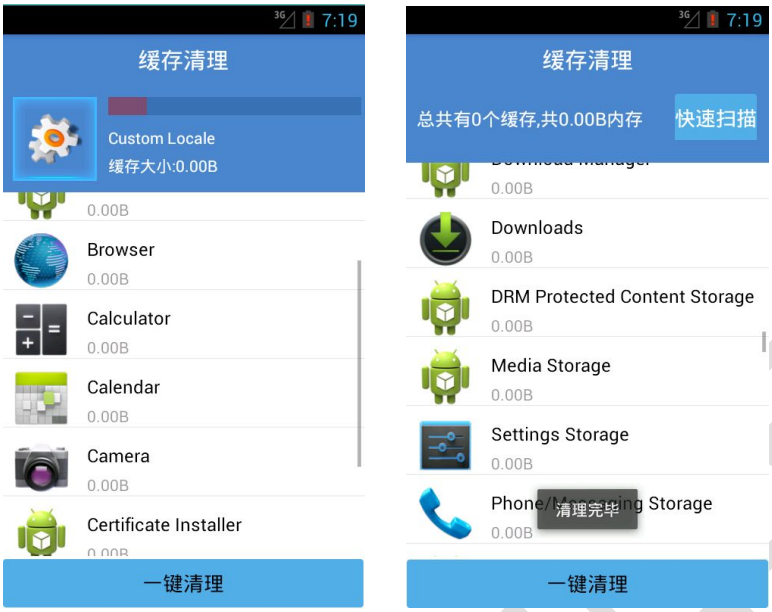

图 8-9 缓冲清理界面

图 8-9 缓存清理界面对应的布局文件如【文件 8-7】所示。

【文件 8-7】 activity_cache.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent"
5.      android:orientation="vertical" >
6.      <TextView
7.          style="@style/TitleBarTextView"
8.          android:gravity="center"
9.          android:text="缓存清理" />
10.     <FrameLayout
11.         android:layout_width="match_parent"
12.         android:layout_height="120dp"
13.         android:background="#429ed6" >
14.         <RelativeLayout
15.             android:id="@+id/rl_content"
16.             android:layout_width="match_parent"
17.             android:layout_height="120dp"
18.             android:background="#429ed6" >
19.             <RelativeLayout
20.                 android:id="@+id/rl_scan_bg"
21.                 android:layout_width="wrap_content"
22.                 android:layout_height="wrap_content"
23.                 android:layout_centerVertical="true"
24.                 android:layout_marginLeft="20dp"
25.                 android:background="@drawable/scan_bg" >
26.                 <ImageView
```

```
27.                 android:id="@+id/iv_icon"
28.                 android:layout_width="50dp"
29.                 android:layout_height="50dp"
30.                 android:layout_centerVertical="true"
31.                 android:layout_marginLeft="15dp"
32.                 android:src="@drawable/ic_launcher" />
33.             <ImageView
34.                 android:id="@+id/iv_scan_line"
35.                 android:layout_width="wrap_content"
36.                 android:layout_height="wrap_content"
37.                 android:layout_alignTop="@id/iv_icon"
38.                 android:layout_marginLeft="6dp"
39.                 android:layout_marginTop="2dp"
40.                 android:src="@drawable/scan_line" />
41.         </RelativeLayout>
42.         <TextView
43.             android:id="@+id/tv_app_name"
44.             android:layout_width="wrap_content"
45.             android:layout_height="wrap_content"
46.             android:layout_centerVertical="true"
47.             android:layout_marginLeft="10dp"
48.             android:layout_marginTop="5dp"
49.             android:layout_toRightOf="@id/rl_scan_bg"
50.             android:singleLine="true"
51.             android:text="联系人"
52.             android:textColor="#fff"
53.             android:textSize="18sp" />
54.         <TextView
55.             android:id="@+id/cache_size"
56.             android:layout_width="wrap_content"
57.             android:layout_height="wrap_content"
58.             android:layout_below="@id/tv_app_name"
59.             android:layout_marginLeft="10dp"
60.             android:layout_marginTop="10dp"
61.             android:layout_toRightOf="@id/rl_scan_bg"
62.             android:text="缓存大小"
63.             android:textColor="#fff"
64.             android:textSize="16sp" />
65.         <ProgressBar
66.             android:id="@+id/progressBar"
67.             style="?android:attr/progressBarStyleHorizontal"
68.             android:layout_width="match_parent"
69.             android:layout_height="wrap_content"
70.             android:layout_alignLeft="@+id/tv_app_name"
71.             android:layout_alignParentRight="true"
```

```
72.              android:layout_alignTop="@+id/rl_scan_bg"
73.              android:layout_marginRight="5dp"
74.              android:progressDrawable="@drawable/progress_horizontal" />
75.       </RelativeLayout>
76.       <LinearLayout
77.          android:id="@+id/ll_content"
78.          android:layout_width="match_parent"
79.          android:layout_height="120dp"
80.          android:background="#429ed6"
81.          android:gravity="center"
82.          android:orientation="horizontal"
83.          android:visibility="invisible">
84.          <TextView
85.             android:id="@+id/tv_cache_count"
86.             android:layout_width="wrap_content"
87.             android:layout_height="wrap_content"
88.             android:textColor="#fff"
89.             android:text="共有多少个缓存.缓存大小 10MB" />
90.          <Button
91.             android:id="@+id/bt_scan"
92.             android:layout_width="wrap_content"
93.             android:layout_height="wrap_content"
94.             android:text="重新扫描" />
95.       </LinearLayout>
96.    </FrameLayout>
97.       <ListView
98.            android:id="@+id/list_view"
99.            android:layout_width="match_parent"
100.           android:layout_height="match_parent"
101.           android:layout_weight="1" >
102.       </ListView>
103.       <Button
104.          android:id="@+id/bt_clean_cache"
105.          android:layout_width="match_parent"
106.          android:layout_height="wrap_content"
107.          android:background="@drawable/dg_btn_confirm_selector"
108.          android:text="一键清理" />
109.    </LinearLayout>
```

8.3.3 缓存清理主逻辑

在清理缓存时，需要在界面上不停的更新数据，显示清理了多少缓存，当缓存清理完成后会在代码中动态切换布局，显示缓存清理完成界面，具体代码如【文件 8-8】所示。

【文件 8-8】 CacheActivity.java

```
1.  public class CacheActivity extends Activity implements OnClickListener {
2.      private ImageView iv_scan_line;
3.      private PackageManager mPm;
4.      // 存放所有的数据
5.      private List<CacheInfo> mDatas;
6.      /**
7.       * 一共有多少个垃圾文件
8.       */
9.      private int cache_count = 0;
10.     /**
11.      * 一共有多少 MB 垃圾文件
12.      */
13.     private int tv_cache_size = 0;
14.     IPackageStatsObserver.Stub mStatsObserver = new IPackageStatsObserver.Stub() {
15.         public void onGetStatsCompleted(PackageStats stats, boolean succeeded) {
16.             try {
17.                 long cacheSize = stats.cacheSize;
18.                 CacheInfo info = new CacheInfo();
19.                 String packageName = stats.packageName;
20.                 PackageInfo packageInfo = mPm.getPackageInfo(packageName, 0);
21.                 Drawable icon = packageInfo.applicationInfo.loadIcon(mPm);
22.                 info.icon = icon;
23.                 String appName = packageInfo.applicationInfo.loadLabel(mPm)
24.                         .toString();
25.                 info.appName = appName;
26.                 info.appPackageName = packageName;
27.                 if (cacheSize > 0) {
28.                     info.cacheSize = cacheSize;
29.                     cache_count++;
30.                     tv_cache_size += cacheSize;
31.                     mDatas.add(0, info);
32.                 } else {
33.                     mDatas.add(info);
34.                 }
35.                 // 更新数据
36.                 task.updateCacheInfo(info);
37.             } catch (Exception e) {
38.                 // TODO Auto-generated catch block
39.                 e.printStackTrace();
40.             }
41.         }
42.     };
43.     private ListView mListView;
44.     private ScanAsyTask task;
```

```
45.     private ImageView iv_icon;
46.     private TextView tv_app_name;
47.     private TextView cache_size;
48.     private ProgressBar progressBar;
49.     private Button bt_clean_cache;
50.     @Override
51.     protected void onCreate(Bundle savedInstanceState) {
52.         // TODO Auto-generated method stub
53.         super.onCreate(savedInstanceState);
54.         // mPm.getPackageSizeInfo(mCurComputingSizePkg, mStatsObserver);
55.         mPm = getPackageManager();
56.         initView();
57.         initData();
58.     }
59.     /**
60.      * 当失去焦点的时候调用
61.      */
62.     @Override
63.     protected void onPause() {
64.         // TODO Auto-generated method stub
65.         super.onPause();
66.         if (null != task) {
67.             // 当正在扫描的时候。如果不等于 null。那么我设置为 null
68.             task.stop();
69.            task = null;
70.         }
71.     }
72.     private class ScanAsyTask extends AsyncTask<Void, CacheInfo, String> {
73.         private CacheAdapter adapter;
74.         private int progress;
75.         /**
76.          * 是否扫描完成
77.          */
78.         boolean isFinish = false;
79.         @Override
80.         protected void onPreExecute() {
81.             // TODO Auto-generated method stub
82.             super.onPreExecute();
83.             // 第一个参数：相对父亲
84.             TranslateAnimation translateAnimation = new TranslateAnimation(
85.                     Animation.RELATIVE_TO_PARENT, 0,
86.                     Animation.RELATIVE_TO_PARENT, 0,
87.                     Animation.RELATIVE_TO_PARENT, 0f,
88.                     Animation.RELATIVE_TO_PARENT, 0.8f);
```

```
89.             translateAnimation.setDuration(800);
90.             translateAnimation.setRepeatCount(Animation.INFINITE);
91.             translateAnimation.setRepeatMode(Animation.REVERSE);
92.             iv_scan_line.startAnimation(translateAnimation);
93.             // 可以点击一键清理
94.             bt_clean_cache.setEnabled(false);
95.         }
96.         /**
97.          * 停止扫描
98.          */
99.         public void stop() {
100.                isFinish = true;
101.            }
102.            /**
103.             * 更新
104.             *
105.             * @param info
106.             */
107.            public void updateCacheInfo(CacheInfo info) {
108.                // 推送进度条
109.                publishProgress(info);
110.            }
111.            @Override
112.            protected void onProgressUpdate(CacheInfo... values) {
113.                // TODO Auto-generated method stub
114.                super.onProgressUpdate(values);
115.                if (isFinish) {
116.                    return;
117.                }
118.                CacheInfo info = values[0];
119.                iv_icon.setImageDrawable(info.icon);
120.                tv_app_name.setText(info.appName);
121.                cache_size.setText(Formatter.formatFileSize(
122.                        getApplicationContext(), info.cacheSize));
123.                // 一键清理可用
124.                bt_clean_cache.setEnabled(true);
125.                progressBar.setProgress(progress);
126.                if (adapter == null) {
127.                    adapter = new CacheAdapter();
128.                    mListView.setAdapter(adapter);
129.                } else {
130.                    adapter.notifyDataSetChanged();
131.                }
132.                // 自己滚动
133.                mListView.smoothScrollToPosition(adapter.getCount());
```

```
134.            }
135.            @Override
136.            protected void onPostExecute(String result) {
137.                // TODO Auto-generated method stub
138.                super.onPostExecute(result);
139.                if (isFinish) {
140.                    return;
141.                }
142.                // 滚动到第 0 个位置
143.                mListView.smoothScrollToPosition(0);
144.                ll_content.setVisibility(View.VISIBLE);
145.                rl_content.setVisibility(View.INVISIBLE);
146.                tv_cache_count.setText("总共有"
147.                        + cache_count
148.                        + "个缓存文件:"
149.                        + Formatter.formatFileSize(getApplicationContext(),
150.                                tv_cache_size) + "缓存文件");
151.            }
152.            @Override
153.            protected String doInBackground(Void... params) {
154.                List<PackageInfo> packages = mPm.getInstalledPackages(0);
155.                // 不可以点击一键清理
156.                progressBar.setMax(packages.size());
157.                progress = 0;
158.                mDatas = new ArrayList<CacheInfo>();
159.                for (PackageInfo packageInfo : packages) {
160.                    String packageName = packageInfo.packageName;
161.                    progress++;
162.                    Drawable icon = packageInfo.applicationInfo.loadIcon(mPm);
163.                    String appName = packageInfo.applicationInfo.loadLabel(mPm)
164.                            .toString();
165.                    try {
166.                        Method method = mPm.getClass().getDeclaredMethod(
167.                                "getPackageSizeInfo", String.class,
168.                                IPackageStatsObserver.class);
169.                        method.invoke(mPm, packageName, mStatsObserver);
170.                        if (isFinish) {
171.                            break;
172.                        }
173.                        SystemClock.sleep(100);
174.                    } catch (Exception e) {
175.                        // TODO Auto-generated catch block
176.                        e.printStackTrace();
177.                    }
```

```
178.                }
179.                return null;
180.            }
181.        }
182.        private class CacheAdapter extends BaseAdapter {
183.            @Override
184.            public int getCount() {
185.                // TODO Auto-generated method stub
186.                return mDatas.size();
187.            }
188.            @Override
189.            public Object getItem(int position) {
190.                // TODO Auto-generated method stub
191.                return mDatas.get(position);
192.            }
193.            @Override
194.            public long getItemId(int position) {
195.                // TODO Auto-generated method stub
196.                return position;
197.            }
198.            @Override
199.            public View getView(int position, View convertView, ViewGroup parent) {
200.                if (convertView == null) {
201.                    convertView = View.inflate(getApplicationContext(),
202.                            R.layout.item_cache_activity, null);
203.                }
204.                ImageView iv_icon = (ImageView) convertView
205.                        .findViewById(R.id.iv_icon);
206.                TextView tv_app_name = (TextView) convertView
207.                        .findViewById(R.id.tv_app_name);
208.                TextView tv_chache_size = (TextView) convertView
209.                        .findViewById(R.id.tv_chache_size);
210.                ImageButton ib_list_button_clean = (ImageButton) convertView
211.                        .findViewById(R.id.ib_list_button_clean);
212.                final CacheInfo cacheInfo = mDatas.get(position);
213.                iv_icon.setImageDrawable(cacheInfo.icon);
214.                tv_app_name.setText(cacheInfo.appName);
215.                tv_chache_size.setText("缓存大小:"
216.                        + Formatter.formatFileSize(getApplicationContext(),
217.                                cacheInfo.cacheSize));
218.            // 判断当前的缓存大小是否大于 0 。如果大于 0.那么就显示清理的图标.如果小于 0.就不显示
219.            ib_list_button_clean.setVisibility(cacheInfo.cacheSize > 0 ? View.VISIBLE
220.                                : View.INVISIBLE);
221.                ib_list_button_clean.setOnClickListener(new OnClickListener() {
222.                    @Override
```

```
223.                    public void onClick(View v) {
224.                        Intent intent = new Intent();
225.                        // <intent-filter>
226.                        // <action
227.                        //android:name="android.settings.APPLICATION_DETAILS_SETTINGS"
228.                        // />
229.                        // <category android:name="android.intent.category.DEFAULT"
230.                        // />
231.                        // <data android:scheme="package" />
232.                        // </intent-filter>
233.                intent.setAction("android.settings.APPLICATION_DETAILS_SETTINGS");
234.                        intent.addCategory("android.intent.category.DEFAULT");
235.                        intent.setData(Uri.parse("package:"
236.                                + cacheInfo.appPackageName));
237.                        startActivity(intent);
238.                    }
239.                });
240.                return convertView;
241.            }
242.        }
243.        private void initData() {
244.            task = new ScanAsyTask();
245.            task.execute();
246.        }
247.        private void initView() {
248.            setContentView(R.layout.activity_cache);
249.            iv_scan_line = (ImageView) findViewById(R.id.iv_scan_line);
250.            mListView = (ListView) findViewById(R.id.list_view);
251.            // 应用的图标
252.            iv_icon = (ImageView) findViewById(R.id.iv_icon);
253.            tv_app_name = (TextView) findViewById(R.id.tv_app_name);
254.            cache_size = (TextView) findViewById(R.id.cache_size);
255.            progressBar = (ProgressBar) findViewById(R.id.progressBar);
256.            // 一键清理
257.            bt_clean_cache = (Button) findViewById(R.id.bt_clean_cache);
258.            bt_clean_cache.setOnClickListener(this);
259.            ll_content = (LinearLayout) findViewById(R.id.ll_content);
260.            ll_content.setVisibility(View.INVISIBLE);
261.            rl_content = (RelativeLayout) findViewById(R.id.rl_content);
262.            rl_content.setVisibility(View.VISIBLE);
263.            // 扫描的个数
264.            tv_cache_count = (TextView) findViewById(R.id.tv_cache_count);
265.            // 重新扫描
266.            Button bt_scan = (Button) findViewById(R.id.bt_scan);
```

```
267.            bt_scan.setOnClickListener(new OnClickListener() {
268.                @Override
269.                public void onClick(View v) {
270.                    // TODO Auto-generated method stub
271.                    initData();
272.                    ll_content.setVisibility(View.INVISIBLE);
273.                    rl_content.setVisibility(View.VISIBLE);
274.                }
275.            });
276.        }
277.        /**
278.         * 一键清理
279.         */
280.        IPackageDataObserver.Stub mDataObserver = new IPackageDataObserver.Stub() {
281.            @Override
282.            public void onRemoveCompleted(String packageName, boolean succeeded)
283.                    throws RemoteException {
284.                Toast.makeText(getApplicationContext(), "清理完毕", 0).show();
285.            }
286.        };
287.        private LinearLayout ll_content;
288.        private RelativeLayout rl_content;
289.        private TextView tv_cache_count;
290.        // @hide
291.        // public abstract void freeStorageAndNotify(long freeStorageSize,
292.        // IPackageDataObserver observer);
293.        @Override
294.        public void onClick(View v) {
295.            try {
296.                Method method = mPm.getClass().getDeclaredMethod(
297.                        "freeStorageAndNotify", long.class,
298.                        IPackageDataObserver.class);
299.                method.invoke(mPm, Long.MAX_VALUE, mDataObserver);
300.                initData();
301.            } catch (Exception e) {
302.                // TODO Auto-generated catch block
303.                e.printStackTrace();
304.            }
305.        }
306.    }
```

- 第 72~181 行是 AsyncTask 的实现类 ScanAsyTask 的主要逻辑，用于开启异步线程去获取手机中的应用信息。
- 第 182~217 行是缓存清理界面的数据适配器，将所有的应用程序的详细信息展示出来。其中第 215 行 Formatter.formatFileSize(getApplicationContext(),cacheInfo.cacheSize)方法将传入的 long 类型值转换为缓存文件的大小和单位。

- 第 218~240 行是用于单个应用程序清理缓存，点击按钮后会调到系统的应用程序信息界面，在该界面上可以直接点击清楚缓存。
- 第 244~277 行的 initData()和 initView()方法，前者是在 onCreate()方法中调用，用于开启一个异步的 AcyncTask 机制，来获取手机中应用程序的缓存信息。后者为初始化布局，在 onCreate()方法中调用。
- 第 278~287 行的获取远程服务代理类的对象，实现了 IpackageDataObserver 接口中的 onRemoveCompleted()方法。
- 第 295~307 行是一键清理所有程序的缓存，通过反射的形式获取到 freeStorageAndNotify(long freeStorageSize, IPackageDataObserver observer)方法，该方法接收两个参数，第一个参数表示要释放的缓存大小，第二个参数是远程服务接口。其中 freeStorageSize 参数比较特殊，假设要释放 2M 缓存，手机有 5M 的内存空间，那么就直接清除 2M 的缓存，假设设置要释放 5M 的缓存，但手机实际只有 2M 的内存空间，那么就只能释放 2M 的缓存。这个 API 可以利用系统漏洞，将这个释放缓存的值设置的非常大，例如要释放 100G 的内存空间，手机只有 2G 内存，而缓存文件占 1G，系统一看即使把缓存文件都删除了也没有无法释放那么多的内存，因此就把缓存文件都清理了，目前市面上的安全软件也都是这样做的。Google 检测这个 API 时没有检测清除缓存权限，因此一般程序可以去用。

需要注意的是，在清除缓存时使用了 IPackageDataObserver.aidl 接口，因此需要将其拷贝到 android.content.pm 包中。运行程序，效果图如图 8-10 所示。

图 8-10　缓存清理主界面完整效果图

8.4 本章小结

本章主要是针对缓存清理模块进行讲解，首先讲解了如何对程序进行扫描，获取到程序中缓存信息，然后讲解了如何清理缓存。