一、刷新按钮的动画效果：

在刷新按钮的位置做两个按钮的布局，一个是原来的静态图片，一个是旋转的动画，然后控制这两个效果的显示和隐藏。

（一）刷新按钮的 style：

1. 在 app/src/main/res/values/ styles.xml 文件里添加一个 style ：

```
<style name="title_update_progressbar_style">
    <item name="android:maxWidth">45dp</item>
    <item name="android:maxHeight">45dp</item>
    <item name="android:indeterminateDrawable">@drawable/title_update_anim</item>
    <item name="android:minWidth">45dp</item>
    <item name="android:minHeight">45dp</item>
</style>
```

2. 在 drawable 文件夹下新建一个 title_update_anim.xml，添加如下内容

注：最后一行 toDegrees 是旋转到哪个角度，这个值越大，旋转速度越快。

```
+ <?xml version="1.0" encoding="utf-8"?>
+ <animated-rotate xmlns:android="http://schemas.android.com/apk/res/android"
+     android:drawable="@drawable/title_update"
+     android:fromDegrees="0.0"
+     android:pivotX="50.0%"
+     android:pivotY="50.0%"
+     android:toDegrees="1080.0"/>
```

（二）改标题栏的布局：

把标题栏布局中右边三个小图标的布局改成下图：

（FrameLayout 布局，一层覆盖一层）

【解释】

FrameLayout 里是刷新按钮的布局，ImageView 是原先的静止的图片，ProgressBar 是旋转动画效果，这两个用 FrameLayout 叠在一起，visibility 属性控制 ProgressBar 默认不显示。用一个 LinearLayout 布局把这三个图标布局到标题栏右侧

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_alignParentRight="true">
    <!--定位按钮-->
    <ImageView
        android:id="@+id/title_location"
        android:layout_width="45.0dip"
        android:layout_height="45.0dip"
        android:src="@drawable/base_action_bar_action_city"/>
    <!--分享按钮-->
    <ImageView
        android:id="@+id/title_share"
        android:layout_width="45.0dip"
        android:layout_height="45.0dip"
        android:src="@drawable/title_share"/>
    <!--刷新按钮-->
    <FrameLayout
        android:id="@+id/title_update"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:layout_alignParentRight="true"
        android:layout_gravity="center">
        <!--静态图片-->
        <ImageView
            android:id="@+id/title_update_btn"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:src="@drawable/title_update"/>
        <!--旋转动画效果-->
        <ProgressBar
            android:id="@+id/title_update_progress"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            style="@style/title_update_progressbar_style"
            android:visibility="gone"/>
    </FrameLayout>
</LinearLayout>
```

以前的布局是这样的：

这个布局方法是先把刷新图标排在最右侧，然后在它左边贴着放一个分享按钮，在分享按钮的左边贴着放一个定位按钮。但是 toLeftOf 对 FrameLayout 就没有那种效果了，所以用一个 LinearLayout

```
<!--刷新图标-->
<ImageView
    android:id="@+id/title_update_btn"
    android:layout_width="45.0dip"
    android:layout_height="45.0dip"
    android:layout_alignParentRight="true"
    android:layout_gravity="center"
    android:src="@drawable/title_update"/>
```

```
<ImageView
    android:id="@+id/title_share"
    android:layout_width="45.0dip"
    android:layout_height="45.0dip"
    android:layout_toLeftOf="@id/title_update_btn"
    android:src="@drawable/title_share"/>
```

```
<ImageView
    android:id="@+id/title_location"
    android:layout_width="45.0dip"
    android:layout_height="45.0dip"
    android:layout_toLeftOf="@id/title_share"
    android:src="@drawable/base_action_bar_action_city"/>
```

（三）在 MainActivity.java 里添加控制刷新按钮静态图片和刷新按钮动画效果来回切换的代码

1. 定义一个 ProgressBar 控件

```
    private ProgressBar mUpdateProgressBar;//刷新按钮动画
```

2. 在 OnCreate 函数里将这个控件与布局文件的 id 关联

```
mUpdateProgressBar = (ProgressBar)findViewById(R.id.title_update_progress);
```

3. 在获取网络数据的函数 QueryWeatherCode()里添加这两句：

隐藏刷新按钮图片，显示刷新动画效果

```
    //使用 获取网络数据 ---weather05
    private void queryWeatherCode(String citycode)
    {
        mUpdateBtn.setVisibility(View.GONE);
        mUpdateProgressBar.setVisibility(View.VISIBLE);

        final String address = "http://wthrcdn.etouch.cn/WeatherApi?citykey=" + citycode;//URL
        Log.d("myWeather",address);
```

【解释】

因为函数 QueryWeatherCode()在点击刷新按钮时调用，点击刷新就切换成刷新动画。

```
//点击刷新按钮
if(view.getId()==R.id.title_update_btn)
{
    //从SharedPreferences中读取城市的id
    /*SharedPreferences（SP）是一种轻量级的数据存储方式,采用Key/value的方式进行映射，最终会在手机的/data/data/pa
    Sp通常用于记录一些参数配置、行为标记等。
    注意：不要使用Sp去存储量大的数据，否则会大大影响应用性能，甚至出现ANR
    # getSharedPreferences(name, mode)获取一个SharedPreferences
    参数1:name在/data/data/package_name/shared_prefs/目录下生成的文件的名字(如果该文件不存在就会创建，如果存在则
    参数2:mode该文件的访问模式(Context.MODE_PRIVATE:默认的创建模式，只能由创建它的或者UID相同的应用程序访问，其余
    */
    SharedPreferences sharedPreferences = getSharedPreferences( name: "config",MODE_PRIVATE);
    String cityCode = sharedPreferences.getString( key: "main_city_code", defValue: "101010100");//从Shared
    Log.d( tag: "myWeather",cityCode);

    //检测是否有网络，如果有就执行“获取网络数据”的函数
    if(NetUtil.getNetworkState((this))!=NetUtil.NETWORN_NONE)
    {
        Log.d( tag: "myWeather", msg: "网络OK");
        queryWeatherCode(cityCode);//获取网络数据
    }else
    {
        Log.d( tag: "myWeather", msg: "网络挂了");
        Toast.makeText( context: MainActivity.this, text: "网络挂了！",Toast.LENGTH_LONG).show();
    }
}
```

4. 在 handleMessage 下面的位置添加这两句：

显示刷新按钮图片，隐藏刷新动画效果

```
public void handleMessage(android.os.Message msg){//覆盖handleMessage方法
    switch (msg.what){//根据收到的消息的what类型处理
        //更新今日天气
        case UPDATE_TODAY_WEATHER:
            updateTodayWeather((TodayWeather) msg.obj);
            mUpdateBtn.setVisibility(View.VISIBLE);
            mUpdateProgressBar.setVisibility(View.GONE);
            break;
        default:
            break;
```

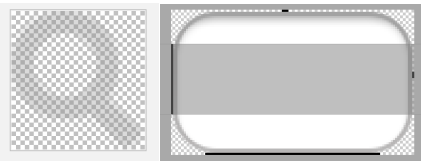
【解释】

在子线程里解析完网络数据之后，通过消息机制，将解析的天气对象发给主线程，主线程接收后调用 updateTodayWeather()来更新界面的天气信息，然后刷新结束，所以在这里添加让刷新按钮恢复原状的代码。

二、搜索框：

（一）搜索框的 UI 布局：

1.资源导入



这两张图，放在 drawable-xhdpi 文件夹里。

2.在 select_city.xml 标题布局的下方，ListView 的上方放搜索框的布局：

```
<!--drawableLeft在文字框左侧添加一个图片；
drawablePadding图片和文字的距离
singleLine单行显示-->
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffbec0c3">
        <EditText
            android:id="@+id/search_edit"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="搜索全国城市（中文/拼音）"
            android:layout_margin="10dp"
            android:drawableLeft="@drawable/magnifying_glass"
            android:drawablePadding="8dp"
            android:paddingBottom="8dp"
            android:paddingLeft="10dp"
            android:paddingRight="30dp"
            android:paddingTop="8dp"
            android:singleLine="true"
            android:background="@drawable/contact_search_box_edittext_keyword_background"
            android:textColor="#ff000000"
            android:textColorHint="#ffcccccc"
            android:textSize="15.0sp" />
    </RelativeLayout>
```

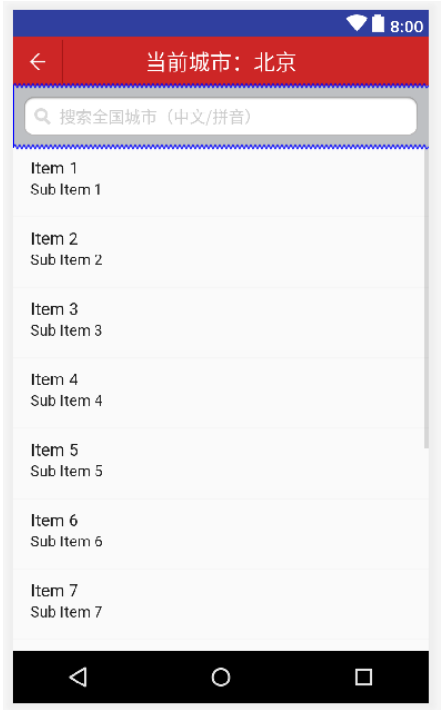
其中：

android:singleLine="true"意思是输入的东西单行显示

#ffbec0c3 是背景那块灰色

textColor 是输入文字的颜色

textColorHint 是 hint 的颜色（搜索全国城市（中文/拼音））



（二）实现搜索功能（用 TextWatcher 的输入监听实现）

1.自定义城市搜索的适配器

新建 `SearchCityAdapter.java`

```
+ public class SearchCityAdapter extends BaseAdapter{
+     private Context mContext;//SelectCity Activity
+     private List<City> mCityLists;//所有城市列表
+     private List<City> mSearchCityLists;//搜索到的城市列表
+     private LayoutInflater mInflater;//布局填充器
+
+     public SearchCityAdapter(Context context, List<City> cityLists){
+         mContext = context;
+         mCityLists = cityLists;
+         mSearchCityLists = new ArrayList<City>();
+         mInflater = LayoutInflater.from(mContext);//从一个Context中，获得一个布局填充器，
+     }
+
```

```
+     @Override
+     //返回ListView里面所有的item子项的个数
+     public int getCount() {
+         return mSearchCityLists.size();
+     }
+
+     @Override
+     //返回每一个item子项
+     public Object getItem(int position) {
+         return mSearchCityLists.get(position);
+     }
+
+     @Override
+     //返回每一个item子项的id
+     public long getItemId(int position) { return position; }
+
+     @Override
+     //返回每一项的显示内容
+     /*第一个参数position----------该视图在适配器数据中的位置
+     第二个参数convertView-----旧视图
+     第三个参数parent-----------此视图最终会被附加到的父级视图*/
+     public View getView(int position, View convertView, ViewGroup parent) {
+         if(convertView == null) {
+             convertView = mInflater.inflate(R.layout.default_search_city,null);//将XML转化为View ///@@@item
+         }
+         TextView provinceTv = (TextView)convertView.findViewById(R.id.search_province);//Item布局中对应的控件===省名
+         TextView cityTv = (TextView)convertView.findViewById(R.id.search_city);//Item布局中对应的控件===城市名
+         provinceTv.setText(mSearchCityLists.get(position).getProvince());//设置控件的属性值-省
+         cityTv.setText(mSearchCityLists.get(position).getCity());//设置控件的属性值-城市名
+         return convertView;
+     }
```

```
+    public Filter getFilter(){
+        Filter filter = new Filter() {
+            @Override
+            /*根据约束条件(CharSequence)调用一个工作线程过滤数据。Filter的子类必须实现该方法来执行过
+            过滤结果以Filter.FilterResults的形式返回，然后在UI线程中通过publishResults(CharSequer
+            约定：当约束条件为null时，原始数据必须被恢复。
+            这里的约束CharSequence constraint是用户在搜索框输入的字符串*/
+            protected FilterResults performFiltering(CharSequence constraint) {
+                String str = constraint.toString().toUpperCase();//获得用户输入的字符串---toUpp
+                FilterResults results = new FilterResults();//本函数返回的过滤结果
+                ArrayList<City> fliterList = new ArrayList<City>();//存储过滤后留下的数据
+
+                if(mCityLists != null && mCityLists.size()!=0){
+                    for(City city : mCityLists){
+                        //getAllFirstPY().indexOf(str): 查找指定字符或字符串str在字符串getAllFir
+                        if(city.getAllFirstPY().indexOf(str) > -1 //全拼音（BEIJING）
+                                || city.getAllPY().indexOf(str) > -1 //每个字的拼音首字母（BJ）
+                                || city.getCity().indexOf(str) > -1){ //城市或区名称（北京）
+                            fliterList.add(city);
+                        }
+                    }
+                }
+                results.values = fliterList;//过滤操作之后的数据的值
+                results.count = fliterList.size();//过滤操作之后的数据的数量
+
+                return results;
+            }
+
+            @Override
+            //通过调用UI线程在用户界面发布过滤结果。
+            // Filter的子类必须实现该方法来显示performFiltering(CharSequence)的过滤结果。
+            protected void publishResults(CharSequence constraint, FilterResults results) {
+                mSearchCityLists = (ArrayList<City>) results.values;
+                if(results.count > 0){
+                    notifyDataSetChanged();//如果适配器的内容改变时,强制调用getView来刷新每个Iten
+                }else {
+                    notifyDataSetInvalidated();//重绘控件（还原到初始状态）
+                }
+            }
+        };
+        return filter;
+    }
```

SearchCityAdapter.java：

```
public class SearchCityAdapter extends BaseAdapter{
    private Context mContext;//SelectCity Activity
    private List<City> mCityLists;//所有城市列表
    private List<City> mSearchCityLists;//搜索到的城市列表
    private LayoutInflater mInflater;//布局填充器

    public SearchCityAdapter(Context context, List<City> cityLists){
        mContext = context;
        mCityLists = cityLists;
        mSearchCityLists = new ArrayList<City>();
```

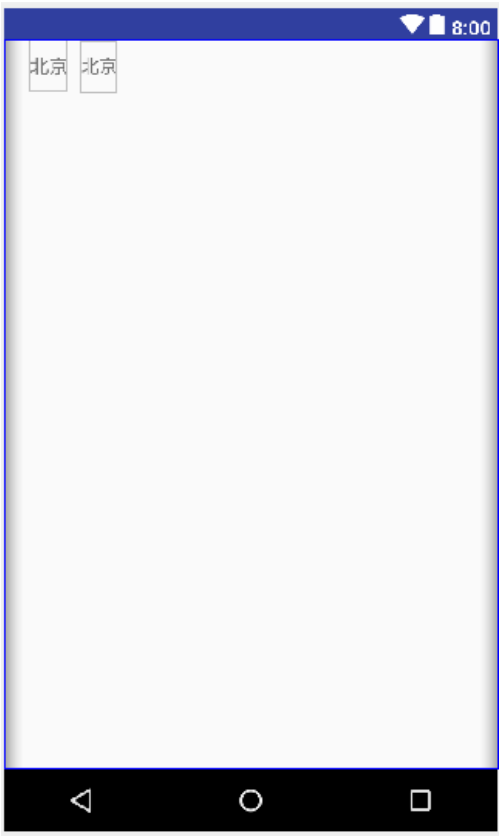
```
        mInflater = LayoutInflater.from(mContext);//从一个 Context 中，获得一个布局填充器，
    }

    @Override
    //返回 ListView 里面所有的 item 子项的个数
    public int getCount() {
        return mSearchCityLists.size();
    }

    @Override
    //返回每一个 item 子项
    public Object getItem(int position) {
        return mSearchCityLists.get(position);
    }

    @Override
    //返回每一个 item 子项的 id
    public long getItemId(int position) { return position; }

    @Override
    //返回每一项的显示内容
    /*第一个参数 position----------该视图在适配器数据中的位置
    第二个参数 convertView-----旧视图
    第三个参数 parent------------此视图最终会被附加到的父级视图*/
    public View getView(int position, View convertView, ViewGroup parent) {
        if(convertView == null) {
            convertView = mInflater.inflate(R.layout.default_search_city,null);// 将 XML 转化为 View ///@@@item
        }
        TextView provinceTv = (TextView)convertView.findViewById(R.id.search_province);//Item 布局中对应的控件===省名
        TextView cityTv = (TextView)convertView.findViewById(R.id.search_city);//Item 布局中对应的控件===城市名
        provinceTv.setText(mSearchCityLists.get(position).getProvince());//设置控件的属性值-省
        cityTv.setText(mSearchCityLists.get(position).getCity());//设置控件的属性值-城市名
        return convertView;
    }

    public Filter getFilter(){
        Filter filter = new Filter() {
```

```
            @Override
            protected FilterResults performFiltering(CharSequence constraint) {
                String str = constraint.toString().toUpperCase();
                FilterResults results = new FilterResults();//本函数返回的过滤结果
                ArrayList<City> fliterList = new ArrayList<City>();//存储过滤后留下的数
据
                if(mCityLists != null && mCityLists.size()!=0){
                    for(City city : mCityLists){
                        //getAllFirstPY().indexOf(str)：查找指定字符或字符串 str 在字符串
getAllFirstPY()中第一次出现地方的索引，未找到的情况返回 -1.
                        if(city.getAllFirstPY().indexOf(str) > -1 //全拼音（BEIJING）
                                || city.getAllPY().indexOf(str) > -1 //每个字的拼音首字母
（BJ）
                                || city.getCity().indexOf(str) > -1){ //城市或区名称（北
京）
                            fliterList.add(city);
                        }
                    }
                }
                results.values = fliterList;//过滤操作之后的数据的值
                results.count = fliterList.size();//过滤操作之后的数据的数量

                return results;
            }

            @Override
            protected  void  publishResults(CharSequence  constraint,  FilterResults
results) {
                mSearchCityLists = (ArrayList<City>) results.values;
                if(results.count > 0){
                    notifyDataSetChanged();
                }else {
                    notifyDataSetInvalidated();
                }
            }
        };
        return filter;
    }
}
```

其中 default_search_city.xml（这是 ListView 中每一项的布局）：

```
<?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="horizontal">
    <TextView
        android:id="@+id/search_province"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dip"
        android:gravity="center_vertical"
        android:minHeight="40.0dip"
        android:text="北京"
        android:textSize="15sp" />
    <TextView
        android:id="@+id/search_city"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dip"
        android:gravity="center_vertical"
        android:minHeight="40.0dip"
        android:text="北京" />
</LinearLayout>
```



2. 在 SelectCity.java 里添加：

```
private EditText mSearchEditText;//搜索框
private SearchCityAdapter mSearchCityAdapter;//搜索城市的适配器
```

```
//初始化UI控件
void InitView() {
    //将控件名与id关联
    titleName=(TextView)findViewById(R.id.title_name);//标题名
    mSearchEditText=(EditText)findViewById(R.id.search_edit);//搜索框
    mSearchEditText.addTextChangedListener(mTextWatcher);
```

使用 TextWatcher 监听 EditText 变化，如果有输入内容就把 ListView 的适配器设置成筛选出搜索结果的适配器，并调用 getFilter 进行数据筛选，否则用原来的适配器。

```
//使用TextWatcher监听EditText变化
TextWatcher mTextWatcher = new TextWatcher() {
    //变化前
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }
    //变化中
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        mSearchCityAdapter = new SearchCityAdapter( context: SelectCity.this, mCityList);//根据输入的内容创建适配器

        mCityListView.setTextFilterEnabled(true);//数据过滤
        if(mCityList.size() < 1 || TextUtils.isEmpty(s)){//输入为空或者城市列表为空
            mCityListView.setAdapter(cityAdapter);
        }else{
            mCityListView.setAdapter(mSearchCityAdapter);//重置ListView的适配器
            mSearchCityAdapter.getFilter().filter(s);//数据的过滤以及刷新展示
        }
    }
    //变化后
    @Override
    public void afterTextChanged(Editable s) {

    }
};
```

改 ListView 适配器的单击事件，ListView 有两个适配器（原先的显示所有城市信息的适配器和当前筛选出搜索结果的适配器），如果当前适配器是搜索的适配器，那么从搜索适配器中获取 item，否则从全部城市列表里获取 item

```
//ListView适配器的单击响应事件
mCityListView.setOnItemClickListener((parent, view, position, id) -> {
        City city;
        if(mSearchCityAdapter != null){
            city = /*mCityList.get(position);*/(City)mSearchCityAdapter.getItem(position);
        }
        else
            city = mCityList.get(position);
        cityCode = city.getNumber();
        updateTitleName(city.getCity());
        Toast.makeText( context: SelectCity.this, text: "你单击了"+position+": "+ city.getCity() +", 编码为"+ cityCode,Toast.LENGTH_SHORT).show();
});
```

SelectCity.java：

```
public class SelectCity extends Activity implements View.OnClickListener{
```

```
    private ImageView mBackBtn;//返回按钮
    private String cityCode; //返回到城市编码
    private TextView titleName;//顶部标题的文字控件
    private EditText mSearchEditText;//搜索框
    private SearchCityAdapter mSearchCityAdapter;//搜索城市的适配器
    private ArrayAdapter<String> cityAdapter;//所有城市的适配器
    private List<City> mCityList;//城市列表
    private ListView mCityListView;//城市管理界面的 ListView

    //使用 TextWatcher 监听 EditText 变化
    TextWatcher mTextWatcher = new TextWatcher() {
        //变化前
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after)
{

        }
        //变化中
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            mSearchCityAdapter = new SearchCityAdapter(SelectCity.this, mCityList);//
根据输入的内容创建适配器

            mCityListView.setTextFilterEnabled(true);//数据过滤
            if(mCityList.size() < 1 || TextUtils.isEmpty(s)){//输入为空或者城市列表为空
                mCityListView.setAdapter(cityAdapter);
            }else{
                mCityListView.setAdapter(mSearchCityAdapter);//重置 ListView 的适配器
                mSearchCityAdapter.getFilter().filter(s);//数据的过滤以及刷新展示
            }
        }
        //变化后
        @Override
        public void afterTextChanged(Editable s) {
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.select_city);

        //更新 ListView
        updateListView();
```

```
        //返回按钮
        mBackBtn = (ImageView)findViewById(R.id.title_back);
        mBackBtn.setOnClickListener(this);

        //更新顶部当前城市文字
        InitView();
    }

    //按钮的单击事件
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            //点击返回按钮
            case R.id.title_back:
                Log.d("myWeather","click back");
                Intent i=new Intent();//weather08-2
                //i.putExtra("cityCode","101160101");
                i.putExtra("cityCode",cityCode);//putExtra 将计算的值回传回去；返回城市编
号--weather08-2
                setResult(RESULT_OK, i);//weather08-2
                finish();//结束当前的 activity 的生命周期
                break;
            default:
                break;
        }
    }

    //用适配器更新 ListView
    void updateListView()
    {
        /*ListView 三种适配器的使用*/
        //--level1
        MyApplication mApplication = MyApplication.getInstance();
        mCityList = mApplication.getCityList();//获取城市列表
        final String[] viewList = new String[mCityList.size()];//显示在 ListView 中的数
据
        int i=0;
        for(City city:mCityList){
            viewList[i]=city.getCity();
            i++;
        }
        mCityListView = (ListView)findViewById(R.id.list_view);
        cityAdapter = new ArrayAdapter<String>(//适配器
                SelectCity.this,
```

```
                android.R.layout.simple_list_item_1,
                viewList);//适配器
        mCityListView.setAdapter(cityAdapter);
        //ListView 适配器的单击响应事件
        mCityListView.setOnItemClickListener(new    AdapterView.OnItemClickListener()
{//单击 ListView 的响应事件
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                City city;
                if(mSearchCityAdapter != null){
                    city                                                                =
/*mCityList.get(position);*/(City)mSearchCityAdapter.getItem(position);
                }
                else
                    city = mCityList.get(position);
                cityCode = city.getNumber();
                updateTitleName(city.getCity());
                Toast.makeText(SelectCity.this,"你单击了"+position+"："+ city.getCity()
+"，编码为"+ cityCode,Toast.LENGTH_SHORT).show();
            }
        });
    }

    //初始化 UI 控件
    void InitView(){
        //将控件名与 id 关联
        titleName=(TextView)findViewById(R.id.title_name);//标题名
        mSearchEditText=(EditText)findViewById(R.id.search_edit);//搜索框
        mSearchEditText.addTextChangedListener(mTextWatcher);

        //初始化控件内容
        titleName.setText("当前城市：北京");
    }

    //更新顶部当前城市的文字
    void updateTitleName(String cityName){
        titleName.setText("当前城市：" + cityName);
    }
}
```


2:56
当前城市：天津
b
北京 北京
北京 八达岭
上海 宝山
天津 宝坻
天津 北辰


2:55
当前城市：天津
bj
北京 北京
山西 太原北郊
陕西 宝鸡
陕西 宝鸡县
青海 铁卜加


2:58
当前城市：天津
beijing
北京 北京

三、Bug 处理：

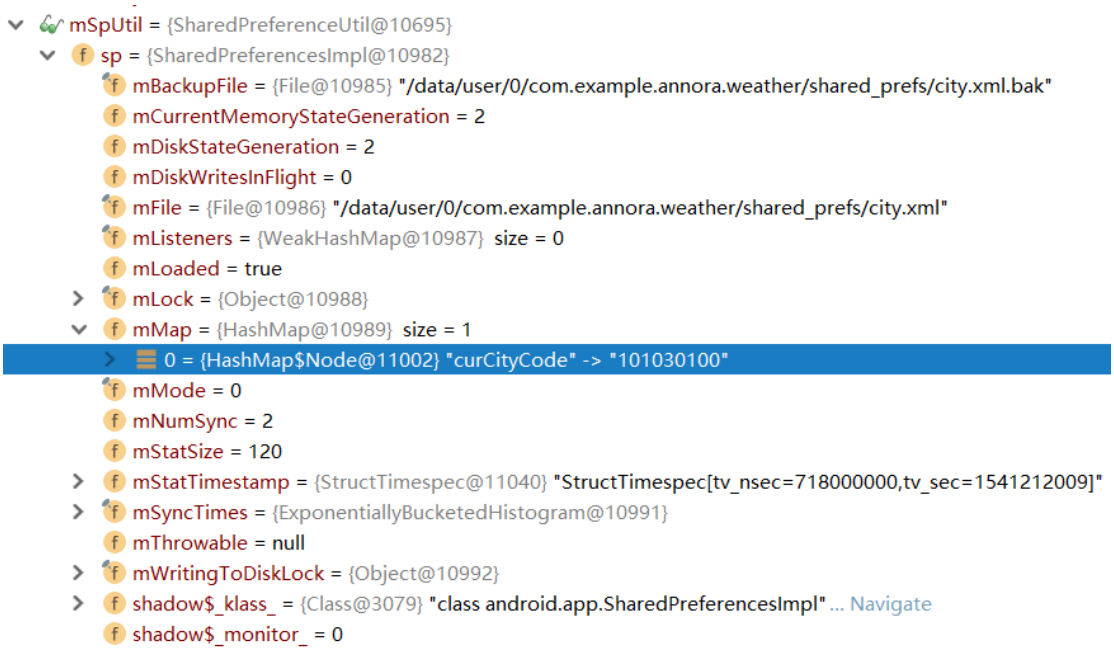
（一）当选择城市界面不选择任何城市，返回主界面异常

处理方式：添加当 SelectCity 界面返回给主界面的 CityCode 为空时的判断，只有不为空的时候才更新天气数据

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    if(requestCode == 1 && resultCode == RESULT_OK) {
        if (data.getStringExtra( name: "cityCode") != null) {//如果选择城市不为空
            String newCityCode = data.getStringExtra( name: "cityCode");//cityCode---SelectCit
            Log.d( tag: "myWeather", msg: "选择的城市代码为" + newCityCode);

            if (NetUtil.getNetworkState( context: this) != NetUtil.NETWORN_NONE) {
                Log.d( tag: "myWeather", msg: "网络OK");
                mSpUtil.setCurCityCode(newCityCode);//选择完城市，设置城市编码
                queryWeatherCode(newCityCode);
            } else {
                Log.d( tag: "myWeather", msg: "网络挂了");
                Toast.makeText( context: MainActivity.this, text: "网络挂了！", Toast.LENGTH_LO
            }
        }
    }
}
```

（二）主界面点击刷新按钮会默认刷新成北京，从主 Activity 进入城市管理界面的时候，标题永远是“当前城市：北京”

处理方式：用 SharedPreference 存储当前城市编码，当进入城市管理界面时获取这个数据更新 title，当选择或定位当前城市，使当前城市改变时，把当前城市存入 SharedPreference



SharedPreference 存储的城市在下次运行模拟器的时候还在

1. 创建 `SharedPreferenceUtil` 工具类：

util/SharedPreferenceUtil.java

```
public class SharedPreferenceUtil {
    public SharedPreferences sp;
    public SharedPreferences.Editor spEditor;
    public static final String CITY_SHAREPRE_FILE = "city";
    public static final String CURR_CITY_CODE = "curCityCode";//当前主界面显示城市的编码

    public SharedPreferenceUtil(Context context, String file){
        sp = context.getSharedPreferences(file, Context.MODE_PRIVATE);
        spEditor = sp.edit();
    }

    //设置当前城市的编码
    public void setCurCityCode(String curCityCode){
        spEditor.putString(CURR_CITY_CODE, curCityCode);//以键值对形式写入 Editor
            spEditor.commit();//提交数据
    }
    //获取当前城市的编码
    public String getCurCityCode(){
        return sp.getString(CURR_CITY_CODE, "");
    }
}
```

2. 在 `MyApplication.java` 中添加：

```
private SharedPreferenceUtil mSpUtil;
```

```
  public void onCreate(){
-30,6 +32,7 @@ public void onCreate(){
        myApplication=this;
        mCityDB=openCityDB();//打开数据库
        initCityList();//初始化城市信息列表
        mSpUtil = new SharedPreferenceUtil(this, SharedPreferenceUtil.CITY_SHAREPRE_FILE);
    }
```

```
public synchronized SharedPreferenceUtil getSharedPreferenceUtil(){
    if(mSpUtil == null){
        mSpUtil = new SharedPreferenceUtil(this, SharedPreferenceUtil.CITY_SHAREPRE_FILE);
    }
    return mSpUtil;
}
```

3. 在 **MainActivity.java** 中添加：

```
    private String mCurCityCode;//当前选择的城市编码
    private SharedPreferenceUtil mSpUtil;
    private MyApplication mApplication;
```

OnCreate()中添加：

```
    mApplication = MyApplication.getInstance();
    mSpUtil = mApplication.getSharedPreferenceUtil();
```

接收 SelectCity 的返回数据后，将选择的城市存入 SharedPreference：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    if(requestCode == 1 && resultCode == RESULT_OK) {
        if (data.getStringExtra( name: "cityCode") != null) {//如果选择城市不为空
            String newCityCode = data.getStringExtra( name: "cityCode");//cityCode---SelectCit
            Log.d( tag: "myWeather", msg: "选择的城市代码为" + newCityCode);

            if (NetUtil.getNetworkState( context: this) != NetUtil.NETWORN_NONE) {
                Log.d( tag: "myWeather", msg: "网络OK");
                mSpUtil.setCurCityCode(newCityCode);//选择完城市，设置城市编码
                queryWeatherCode(newCityCode);
            } else {
                Log.d( tag: "myWeather", msg: "网络挂了");
                Toast.makeText( context: MainActivity.this, text: "网络挂了！", Toast.LENGTH_LO
            }
        }
    }
}
```

点击刷新按钮时，如果 SharedPreference 里存的城市编码为空，就默认为北京，并存入 SharedPreference；否则从 SharedPreference 中获取城市编码，刷新网络数据。

```
//点击刷新按钮
if(view.getId()==R.id.title_update_btn)
{

    //检测是否有网络，如果有就执行“获取网络数据”的函数
    if(NetUtil.getNetworkState((this))!=NetUtil.NETWORN_NONE)
    {
        Log.d( tag: "myWeather", msg: "网络OK");

          SharedPreferences sharedPreferences = getSharedPreferences("config", MODE_PRIVATE);
          String cityCode = sharedPreferences.getString("main_city_code", "101010100");//从SharedPr
          Log.d("myWeather", cityCode + "没有选择默认为北京");

        String cityCode;
        //如果存在SharedPreferences的城市编码为空，就默认设置为北京
        if(TextUtils.isEmpty(mSpUtil.getCurCityCode())){
            mSpUtil.setCurCityCode("101010100");//把默认城市北京保存到SharedPreferences
            cityCode = mSpUtil.getCurCityCode();
            Log.d( tag: "SP", msg: "获取默认编码为：" + cityCode);
        }else {
            cityCode = mSpUtil.getCurCityCode();//获取城市编码
            Log.d( tag: "SP", msg: "获取编码为：" + cityCode);
            queryWeatherCode(cityCode);//获取网络数据
        }
    }else
    {
        Log.d( tag: "myWeather", msg: "网络挂了");
        Toast.makeText( context: MainActivity.this, text: "网络挂了！",Toast.LENGTH_LONG).show();
    }
}
```

4. 在 **SelectCity.java** 中添加：

（SelectCity.java 中应该在 OnCreate 时从 SharedPreference 中获取当前城市编码，然后对应找到城市名，并用它更新当前城市 title。）

这里的 HashMap 用来存城市编码-城市信息的键值对，这样根据城市编码就能查到一条城市的信息，然后可以获取城市名来更新 title

```
private SharedPreferenceUtil mSpUtil;
private MyApplication mApplication;
private HashMap<String, City> cityCode_cityHashMap;
```

在 OnCreate()中：

```
mApplication = MyApplication.getInstance();
mCityList = mApplication.getCityList();//获取城市列表
mSpUtil = mApplication.getSharedPreferenceUtil();
cityCode_cityHashMap = new HashMap<>();
for(City city : mCityList){
    cityCode_cityHashMap.put(city.getNumber(), city);
}
```

```
 //titleName.setText("当前城市：北京");
 City curCity = cityCode_cityHashMap.get(mSpUtil.getCurCityCode());//根据当前城市编码得到城市名
 titleName.setText("当前城市：" + curCity.getCity());
```

（三）部分城市缺数据

找一个缺数据的小城市，查看 parseXML()函数返回的数据

这个数据通过消息机制传给 updateTodayWeather()处理

```
case UPDATE_TODAY_WEATHER:
    updateTodayWeather((TodayWeather) msg.obj);  msg: "{ when=-396ms what=1 obj=TodayWeather{city='莱阳', updatetime='11:56', wendu='18'
    mUpdateBtn.setVisibility(View.VISIBLE);
    mUpdateProgressBar.setVisibility(View.GONE);
    break;
```

1.在 updateTodayWeather()根据网络数据刷新 TextView 的前面添加信息为 null 时的处理：

```
todayWeather.missData();//将缺失的数据设置为
```

2.在 TodayWeather.java 中将无数据的信息赋值

```
public void missData(){
    city = setMissData(city);
    updatetime = setMissData(updatetime);
    wendu = setMissData(wendu);
    shidu = setMissData(shidu);
    pm25 = setMissData(pm25);
    quality = setMissData(quality);
    fengxiang = setMissData(fengxiang);
    fengli = setMissData(fengli);
    date = setMissData(date);
    high = setMissData(high);
    low = setMissData(low);
    type = setMissData(type);
}
public String setMissData(String data){
    return data == null? "无数据": data;
}
```

————————————————

PS：还有一个遇到的问题：

运行模拟器时电脑突然关机=_=。。。再打开就一直 build failed

```
Error:Execution failed for task ':app:transformClassesWithInstantRunForDebug'.
java.lang.IllegalStateException: Expected BEGIN_ARRAY but was STRING at line 1 column 1
path $
```

解决方式：

Yeah It's Just a bug in Android Studio, try `Build -> Clean Project`, and after that click `Build -> Rebuild Project`