目 录

设计显示天气代码

本章使用 GSON 解析天气信息。

1. 定义 GSON 实体类

根据和天气返回的 json 格式的数据，筛选我们需要是重要数据，大体格式如下：

```
{
    "HeWeather":[
    {
        "status":"ok",
        "basic":{},
        "aqi":{},
        "now":{},
        "suggestion":{},
        "daily_forecast":[]
    }
    ]
}
```

其中，basic、aqi、now、suggestion 和 daily_forecast 的内容又有具体的内容，将这 5 个部分定义成 5 个实体类。下面分别分析这 5 部分的组成。

（1）basic:

```
"basic":{
        "city":"济南",
        "id":"CN101120101",
        "update":{"loc":"2018-08-22 15:45"}
      }
```

city 表示城市名，id 表示城市对应的天气 id，update 中的 loc 表示天气的更新时间。按照此结构在 gson 包下建立 Basic 类，代码如下：

```
package com.coolweather.android.gson;
import com.google.gson.annotations.SerializedName;
```

```
public class Basic {
    @SerializedName("city")
    public String cityName;
    @SerializedName("id")
    public String weatherId;
    public Update update;
    public class Update {
        @SerializedName("loc")
        public String updateTime;
    }
}
```

json 中一些字段可能不适合直接作为 Java 字段命名，因此这里使用了@SerializedName 注解的方式让 json 字段和 java 字段之间建立映射关系。

（2）aqi 类

```
"aqi":{
    "city":{
        "aqi":"73",
        "pm25":"53",
    }
}
```

在 gson 包下建 AQI 类，代码如下：

```
package com.coolweather.android.gson;
public class AQI {
    public AQICity city;
    public class AQICity {
        public String aqi;
        public String pm25;
    }
}
```

（3）now 类

```
"now":{
    "tmp":"24",
    "cond":{
        "txt":"阴"
        }
    }
```

在 gson 包下建 now 类，代码如下：

```
package com.coolweather.android.gson;
import com.google.gson.annotations.SerializedName;
public class Now {
    @SerializedName("tmp")
    public String temperature;
    @SerializedName("cond")
    public More more;
    public class More {
        @SerializedName("txt")
        public String info;
    }
}
```

（4）suggestion 类

```
"suggestion":{
    "comf":{
        "txt":"白天以阴或多云天气为主，但稍会让您感到有点儿热，但大部分
    人完全可以接受。"},
    "sport":{
        "txt":"阴天，且天气较热，请减少运动时间并降低运动强度。"},
    "cw":{
        "txt":"较适宜洗车，未来一天无雨，风力较小，擦洗一新的汽车至少能
```

保持一天。"}

}

在 gson 包下新建一个 Suggestion 类，代码如下：

```
package com.coolweather.android.gson;
import com.google.gson.annotations.SerializedName;
public class Suggestion {
    @SerializedName("comf")
    public Comfort comfort;
    @SerializedName("cw")
    public CarWash carWash;
    public Sport sport;
    public class Comfort {
        @SerializedName("txt")
        public String info;
    }
    public class CarWash {
        @SerializedName("txt")
        public String info;
    }
    public class Sport {
        @SerializedName("txt")
        public String info;
    }
}
```

（5）daily_forecast 类

```
"daily_forecast":[
    {
        "date":"2018-08-22",
        "cond":{
```

```
            "txt_d":"阴"
            },
            "tmp":{
                "max":"30","
                min":"22"
                }
            },
            {
                "date":"2018-08-23",
                "cond":{
                    "txt_d":"晴"
                    },
                "tmp":{
                    "max":"30",
                    "min":"20"
                    }
                },
        }
```

Daily_forecast 中包含的是一个数组,数组中的每一项都代表着未来一天的天气信息，针对这种情况，定义单日天气的实体类，然后在声明实体类引用的时候使用集合类型进行声明。

```
package com.coolweather.android.gson;
import com.google.gson.annotations.SerializedName;
public class Forecast {
    public String date;
    @SerializedName("tmp")
    public Temperature temperature;
    @SerializedName("cond")
    public More more;
```

```
    public class Temperature {
        public String max;
        public String min;
    }
    public class More {
        @SerializedName("txt_d")
        public String info;
    }
}
```

（6）Weather 类，代码如下：

```
package com.coolweather.android.gson;
import com.google.gson.annotations.SerializedName;
import java.util.List;
public class Weather {
    public String status;
    public Basic basic;
    public AQI aqi;
    public Now now;
    public Suggestion suggestion;
    @SerializedName("daily_forecast")
    public List<Forecast> forecastList;
}
```

在 Weather 类中，对 Basic、AQI、Now、Suggestion 和 Forecast 类进行了引用，其中由于 daily_forecast 中包含的是一个数组，因此在这里使用了 List 集合来引用 Forecast 类。

返回的天气数据中包含一项 status 数据，成功返回 ok，失败返回具体的原因，那么这里也添加了一个对应的 status 字段。

2. 编写天气界面

首先创建一个由于显示天气信息的活动。右击 com.coolweather.android 包，

new，activity，epty activity，创建一个 WeatherActivity，将布局名制定为 activity_weather.xml。

为了减少 activity_weather.xml 长度，我们将界面的不同部分写在不同的布局文件里，再通过引入布局的方式集成到 activity_weather.xml 中。

在 Layout resource file 中新建 title.xml 作为头布局，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize">
    <Button
        android:id="@+id/nav_button"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_marginLeft="10dp"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:background="@drawable/ic_home" />
    <TextView
        android:id="@+id/title_city"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textColor="#fff"
        android:textSize="20sp" />
    <TextView
        android:id="@+id/title_update_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginRight="10dp"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:textColor="#fff"
        android:textSize="16sp"/>
</RelativeLayout>
```

本布局文件中两个 TextView，一个居中显示城市名，一个居右显示更新时间。

新建 now.xml 作为当前天气信息的布局，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp">
    <TextView
        android:id="@+id/degree_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end"
        android:textColor="#fff"
        android:textSize="60sp" />
    <TextView
        android:id="@+id/weather_info_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end"
        android:textColor="#fff"
```

```
        android:textSize="20sp" />
</LinearLayout>
```

本文件中两个 TextView，一个用于显示当前气温，一个用于显示天气概况。

新建一个 forecast.xml 文件，作为未来几天天气信息的布局，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:background="#8000">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:layout_marginTop="15dp"
        android:text="预报"
        android:textColor="#fff"
        android:textSize="20sp"/>
    <LinearLayout
        android:id="@+id/forecast_layout"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </LinearLayout>
</LinearLayout>
```

本文件最外层使用了 LinearLayout 定义了一个半透明的背景，然后使用 TextView 定义一个标题，使用一个 LinearLayout 定义一个用于显示未来几天天

气信息的布局，这个布局没有任何内容，因为这是要根据服务器返回的数据在代码中动态添加。

定义一个未来天气的子项布局，创建 forecast_item.xml 文件，代码如下：

```
<?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="wrap_content"
    android:layout_margin="15dp">
    <TextView
        android:id="@+id/date_text"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="2"
        android:textColor="#fff"/>
    <TextView
        android:id="@+id/info_text"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:gravity="center"
        android:textColor="#fff"/>
    <TextView
        android:id="@+id/max_text"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
        android:layout_weight="1"
        android:gravity="right"
        android:textColor="#fff"/>
    <TextView
        android:id="@+id/min_text"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="right"
        android:textColor="#fff"/>

</LinearLayout>
```

本文件中 4 个 TextView，一个用于显示天气预报日期，一个用于显示天气概况，另外两个分别用于显示当天的最高温度和最低温度。

新建 aqi.xml 作为空气质量信息的布局，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:background="#8000">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:layout_marginTop="15dp"
```

```
        android:text="空气质量"
        android:textColor="#fff"
        android:textSize="20sp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="15dp">
        <RelativeLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1">
           <LinearLayout
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_centerInParent="true">
                <TextView
                    android:id="@+id/aqi_text"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center"
                    android:textColor="#fff"
                    android:textSize="40sp"
                    />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center"
                    android:text="AQI 指数"
```

```
            android:textColor="#fff"/>
        </LinearLayout>
    </RelativeLayout>
    <RelativeLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true">
            <TextView
                android:id="@+id/pm25_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:textColor="#fff"
                android:textSize="40sp"
                />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="PM2.5 指数"
                android:textColor="#fff"
                />
        </LinearLayout>
    </RelativeLayout>
```

```
    </LinearLayout>
</LinearLayout>
```

本文件中使用一个 LinearLayout 定义一个半透明的背景，然后使用 TextView 定义一个标题，再使用 LinearLayout 和 RelativeLayout 嵌套的方式实现了一个左右平分且居中对齐的布局，分别用于显示 AQI 指数和 PM2.5 指数。

新建 suggestion.xml 作为生活建议信息的布局，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:background="#8000">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:layout_marginTop="15dp"
        android:text="生活建议"
        android:textColor="#fff"
        android:textSize="20sp"/>
    <TextView
        android:id="@+id/comfort_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:textColor="#fff" />
    <TextView
```

```
        android:id="@+id/car_wash_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:textColor="#fff" />
    <TextView
        android:id="@+id/sport_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:textColor="#fff" />
</LinearLayout>
```

本文件中先定义一个半透明的背景和一个标题,然后下面使用 3 个 TextView 分别用于显示舒适度、洗车指数和运动建议的相关数据。

最后把每个部分的布局文件编好后,将它们引入到 activity_weather.xml 中,如下所示:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary">
    <ImageView
        android:id="@+id/bing_pic_img"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop" />
    <android.support.v4.widget.DrawerLayout
        android:id="@+id/drawer_layout"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent">
<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/swipe_refresh"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ScrollView
        android:id="@+id/weather_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="none"
        android:overScrollMode="never">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:fitsSystemWindows="true">
            <include layout="@layout/title" />
            <include layout="@layout/now" />
            <include layout="@layout/forecast" />
            <include layout="@layout/aqi" />
            <include layout="@layout/suggestion" />
        </LinearLayout>
    </ScrollView>
</android.support.v4.widget.SwipeRefreshLayout>
<fragment
    android:id="@+id/choose_area_fragment"
    android:name="com.coolweather.android.ChooseAreaFragment"
    android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:layout_gravity="start"
            />
    </android.support.v4.widget.DrawerLayout>
</FrameLayout>
```

本文件中首先最外层布局使用了一个 FrameLayout，并将它的背景色设置成 colorPrimary，然后在 Framelayout 中嵌套一个 ScrollView，这是因为天气界面中的内容比较多，使用 ScrollView 可以允许我们通过滚动的方式查看屏幕以外的内容。

由于 ScrollView 的内部只允许存在一个直接子布局，因此这里又嵌套了一个垂直方向的 LinearLayout，然后在 LinearLayout 中将刚才定义的所有布局逐个引入。

3. 将天气显示到界面上

（1）在 Utility 类中添加一个解析天气的 json 数据的方法，如下所示：

```
/**
 * 将返回的 JSON 数据解析成 Weather 实体类
 */
public static Weather handleWeatherResponse(String response) {
    try {
        JSONObject jsonObject = new JSONObject(response);
        JSONArray jsonArray = jsonObject.getJSONArray("HeWeather");
        String weatherContent = jsonArray.getJSONObject(0).toString();
        return new Gson().fromJson(weatherContent, Weather.class);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

本方法中先是通过 JSONObject 和 JSONArray 将天气数据中的主体内容解析

出来，然后由于之前已经按照上面的数据格式定义过相应的 GSON 实体类，因此只需要通过 fromJson（）方法就能直接将 JSON 数据转换成 Weather 对象。

（2）在活动中去请求天气数据，以及将数据展示到界面上，修改 WeatherActivity 中的代码，如下所示：

```
package com.coolweather.android;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
import com.bumptech.glide.Glide;
import com.coolweather.android.gson.Forecast;
import com.coolweather.android.gson.Weather;
import com.coolweather.android.service.AutoUpdateService;
import com.coolweather.android.util.HttpUtil;
import com.coolweather.android.util.Utility;
```

```
import java.io.IOException;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.Response;
public class WeatherActivity extends AppCompatActivity {
    public DrawerLayout drawerLayout;
    public SwipeRefreshLayout swipeRefresh;
    private ScrollView weatherLayout;
    private Button navButton;
    private TextView titleCity;
    private TextView titleUpdateTime;
    private TextView degreeText;
    private TextView weatherInfoText;
    private LinearLayout forecastLayout;
    private TextView aqiText;
    private TextView pm25Text;
    private TextView comfortText;
    private TextView carWashText;
    private TextView sportText;
    private ImageView bingPicImg;
    private String mWeatherId;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (Build.VERSION.SDK_INT >= 21) {
            View decorView = getWindow().getDecorView();

decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
| View.SYSTEM_UI_FLAG_LAYOUT_STABLE);
```

```
            getWindow().setStatusBarColor(Color.TRANSPARENT);
        }
        setContentView(R.layout.activity_weather);
        // 初始化各控件
        bingPicImg = (ImageView) findViewById(R.id.bing_pic_img);
        weatherLayout = (ScrollView)
findViewById(R.id.weather_layout);
        titleCity = (TextView) findViewById(R.id.title_city);
        titleUpdateTime = (TextView)
findViewById(R.id.title_update_time);
        degreeText = (TextView) findViewById(R.id.degree_text);
        weatherInfoText = (TextView)
findViewById(R.id.weather_info_text);
        forecastLayout = (LinearLayout)
findViewById(R.id.forecast_layout);
        aqiText = (TextView) findViewById(R.id.aqi_text);
        pm25Text = (TextView) findViewById(R.id.pm25_text);
        comfortText = (TextView) findViewById(R.id.comfort_text);
        carWashText = (TextView) findViewById(R.id.car_wash_text);
        sportText = (TextView) findViewById(R.id.sport_text);
        swipeRefresh = (SwipeRefreshLayout)
findViewById(R.id.swipe_refresh);
        swipeRefresh.setColorSchemeResources(R.color.colorPrimary);
        drawerLayout = (DrawerLayout)
findViewById(R.id.drawer_layout);
        navButton = (Button) findViewById(R.id.nav_button);
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
        String weatherString = prefs.getString("weather", null);
```

```
        if (weatherString != null) {
            // 有缓存时直接解析天气数据
            Weather weather =
Utility.handleWeatherResponse(weatherString);
            mWeatherId = weather.basic.weatherId;
            showWeatherInfo(weather);
        } else {
            // 无缓存时去服务器查询天气
            mWeatherId = getIntent().getStringExtra("weather_id");
            weatherLayout.setVisibility(View.INVISIBLE);
            requestWeather(mWeatherId);
        }
        swipeRefresh.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                requestWeather(mWeatherId);
            }
        });
        navButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                drawerLayout.openDrawer(GravityCompat.START);
            }
        });
        String bingPic = prefs.getString("bing_pic", null);
        if (bingPic != null) {
            Glide.with(this).load(bingPic).into(bingPicImg);
        } else {
```

```
            loadBingPic();
        }
    }
    /**
     * 根据天气 id 请求城市天气信息。
     */
    public void requestWeather(final String weatherId) {
        String weatherUrl = "http://guolin.tech/api/weather?cityid=" +
weatherId + "&key=bc0418b57b2d4918819d3974ac1285d9";
        HttpUtil.sendOkHttpRequest(weatherUrl, new Callback() {
            @Override
            public void onResponse(Call call, Response response)
throws IOException {
                final String responseText = response.body().string();
                final Weather weather =
Utility.handleWeatherResponse(responseText);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (weather != null &&
"ok".equals(weather.status)) {
                            SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(WeatherActivity.this).e
dit();
                            editor.putString("weather", responseText);
                            editor.apply();
                            mWeatherId = weather.basic.weatherId;
                            showWeatherInfo(weather);
                        } else {
```

```
                        Toast.makeText(WeatherActivity.this, "获取天气信息失败", Toast.LENGTH_SHORT).show();
                    }
                    swipeRefresh.setRefreshing(false);
                }
            });
        }
        @Override
        public void onFailure(Call call, IOException e) {
            e.printStackTrace();
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(WeatherActivity.this, "获取天气信息失败", Toast.LENGTH_SHORT).show();
                    swipeRefresh.setRefreshing(false);
                }
            });
        }
    });
    loadBingPic();
}
/**
 * 加载必应每日一图
 */
private void loadBingPic() {
    String requestBingPic = "http://guolin.tech/api/bing_pic";
    HttpUtil.sendOkHttpRequest(requestBingPic, new Callback() {
        @Override
```

```
            public void onResponse(Call call, Response response)
throws IOException {
                final String bingPic = response.body().string();
                SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(WeatherActivity.this).e
dit();
                editor.putString("bing_pic", bingPic);
                editor.apply();
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {

Glide.with(WeatherActivity.this).load(bingPic).into(bingPicImg);
                    }
                });
            }
            @Override
            public void onFailure(Call call, IOException e) {
                e.printStackTrace();
            }
        });
    }
    /**
     * 处理并展示 Weather 实体类中的数据。
     */
    private void showWeatherInfo(Weather weather) {
        String cityName = weather.basic.cityName;
        String updateTime = weather.basic.update.updateTime.split("
")[1];
```

```
        String degree = weather.now.temperature + "℃";
        String weatherInfo = weather.now.more.info;
        titleCity.setText(cityName);
        titleUpdateTime.setText(updateTime);
        degreeText.setText(degree);
        weatherInfoText.setText(weatherInfo);
        forecastLayout.removeAllViews();
        for (Forecast forecast : weather.forecastList) {
            View view =
LayoutInflater.from(this).inflate(R.layout.forecast_item,
forecastLayout, false);
            TextView dateText = (TextView)
view.findViewById(R.id.date_text);
            TextView infoText = (TextView)
view.findViewById(R.id.info_text);
            TextView maxText = (TextView)
view.findViewById(R.id.max_text);
            TextView minText = (TextView)
view.findViewById(R.id.min_text);
            dateText.setText(forecast.date);
            infoText.setText(forecast.more.info);
            maxText.setText(forecast.temperature.max);
            minText.setText(forecast.temperature.min);
            forecastLayout.addView(view);
        }
        if (weather.aqi != null) {
            aqiText.setText(weather.aqi.city.aqi);
            pm25Text.setText(weather.aqi.city.pm25);
        }
```

```
            String comfort = "舒适度: " + weather.suggestion.comfort.info;
            String carWash = "洗车指数:" + weather.suggestion.carWash.info;
            String sport = "运行建议: " + weather.suggestion.sport.info;
            comfortText.setText(comfort);
            carWashText.setText(carWash);
            sportText.setText(sport);
            weatherLayout.setVisibility(View.VISIBLE);
            Intent intent = new Intent(this, AutoUpdateService.class);
            startService(intent);
        }
    }
```

在 onCreate()方法中仍然先是获取一些控件的实例，然后会尝试从本地缓存中读取天气数据，如果是第一次是没有缓存，因此就会从网上取出天气 id，并调用 requestWeather()方法来从服务器请求天气数据。注意请求数据的时候先将 ScrollView 进行隐藏，不然空数据的界面看上去会很奇怪。

requestWeather()方法中先是使用了参数中传入的天气 id 和知情权申请好的 API Key 拼装出一个接口地址，接着调用 HttpUtil.sendOkHttpRequest()方法来向该地址发出，服务器会将相应城市的天气信息以 JSON 格式返回。

然后在 onResponse()方法中回调中先调用 Utility.handleWeatherResponse()方法将返回的 Json 格式数据转换成 Weather 对象，再将当前线程切换到主线程，然后进行判断如果服务器返回的 status 状态是 ok，就说明请求天气成功了，此时将返回的数据缓存到 SharedPreferences 当中，并调用 showWeatherInfo()方法进行内容显示。

showWeatherInfo()方法中的逻辑比较简单，其实就是从 Weather 对象中获取数据，然后显示到相应的控件上。注意在未来几天天气预报的部分使用了一个 for 循环来处理每天的天气信息，在循环中动态加载 forecast_item.xml 布局并设置相应的数据，然后添加到父布局当中，设置完所有数据之后，将 ScrollView 重新变成可见。

当下一次再进入 WeatherActivity 时，由于缓存已经存在，因此会直接解析

并显示天气数据，不会再发起网络请求。

（3）从省市县列表跳转到天气界面，修改 ChooseAreaFragment 代码，如下所示：

```
@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            if (currentLevel == LEVEL_PROVINCE) {
                selectedProvince = provinceList.get(position);
                queryCities();
            } else if (currentLevel == LEVEL_CITY) {
                selectedCity = cityList.get(position);
                queryCounties();
            } else if (currentLevel == LEVEL_COUNTY) {
                String                    weatherId                    =
countyList.get(position).getWeatherId();
                if (getActivity() instanceof MainActivity) {
                    Intent    intent    =    new    Intent(getActivity(),
WeatherActivity.class);
                    intent.putExtra("weather_id", weatherId);
                    startActivity(intent);
                    getActivity().finish();
                } else if (getActivity() instanceof WeatherActivity)
{
                    WeatherActivity    activity    =    (WeatherActivity)
```

```
getActivity();
                    activity.drawerLayout.closeDrawers();
                    activity.swipeRefresh.setRefreshing(true);
                    activity.requestWeather(weatherId);
                }
            }
        }
    });
```

在onItemClick()方法中加入了一个if判断,如果当前级别是LEVEL_COUNTY,就启动WeatherActivity，并把当前选中县的天气id传递过去。

（4）缓存数据

在MainActivity中加入一个缓存数据的判断，修改代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
    if (prefs.getString("weather", null) != null) {
        Intent intent = new Intent(this, WeatherActivity.class);
        startActivity(intent);
        finish();
    }
}
```

onCreate（）方法先从SharedPreferences文件中读取缓存数据，如果不为null就说明之前已经请求过天气数据，那么就没有必要让用户再次选择城市，而是直接跳转到WeatherActivity即可。

运行程序，如下图所示：

向下滑动查看更多天气信息，如下图所示：

4. 获取必应每日一图

使用技术让不同的城市或天气情况展示不同的背景图片，访问：http://guolin.tech/api/bing_pic，服务器返回今日必应的背景图链接：http://cn.bing.com/az/hprichbg/rb/FranceMenton_ZH-CN8996032014_1920x1080.jpg

使用 Glide 去加载这张图片就可以了。

修改 activity_weather.xml 代码，如下所示：

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary">
    <ImageView
        android:id="@+id/bing_pic_img"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop" />
……
</FrameLayout>
```

添加一个 ImageView，并且将它的宽和高都设置成 match_parent，由于 FrameLayout 默认情况下会将控件都放置在左上角，因此 ScrollView 会完全覆盖住 ImageView，从而 ImageView 也就成为背景图片。

接着修改 WeatherActivity 中的代码，如下所示：

```
public class WeatherActivity extends AppCompatActivity {
private ImageView bingPicImg;
bingPicImg = (ImageView) findViewById(R.id.bing_pic_img);
……..
String bingPic = prefs.getString("bing_pic", null);
if (bingPic != null) {
```

```
    Glide.with(this).load(bingPic).into(bingPicImg);
} else {
    loadBingPic();
}
/**
 * 根据天气 id 请求城市天气信息。
 */
public void requestWeather(final String weatherId) {
.........
loadBingPic();
}
/**
 * 加载必应每日一图
 */
private void loadBingPic() {
    String requestBingPic = "http://guolin.tech/api/bing_pic";
    HttpUtil.sendOkHttpRequest(requestBingPic, new Callback() {
        @Override
        public void onResponse(Call call, Response response) throws
IOException {
            final String bingPic = response.body().string();
            SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(WeatherActivity.this).e
dit();
            editor.putString("bing_pic", bingPic);
            editor.apply();
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
```

```
Glide.with(WeatherActivity.this).load(bingPic).into(bingPicImg);
                }
            });
        }

        @Override
        public void onFailure(Call call, IOException e) {
            e.printStackTrace();
        }
    });
}
```

onCreate（）方法中获取新增控件 ImageView 的实例，然后尝试从 SharedPreferences 中读取缓存的背景图片，如果有缓存的话就直接使用 Glide 来加载这张图片，如果没有的话就调用 loadBingPic（）方法去请求今日的必应背景图。

loadBingPic（）方法先调用 HttpUtil.sendOKHttpRequest（）方法获取到必应背景图的链接，然后将这个链接缓存到 SharedPreferences 中，再将当前线程切换到主线程，最后使用 Glide 来加载这张图片就可以了。

在 requestWeather（）方法的最后需要调用一下 loadBingPic（）方法，这样在每次请求天气信息的时候同时也会刷新背景图片。运行程序效果如下：

图片

下面将背景图片和状态栏融合在一起，代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (Build.VERSION.SDK_INT >= 21) {
        View decorView = getWindow().getDecorView();
```

```
decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
| View.SYSTEM_UI_FLAG_LAYOUT_STABLE);
            getWindow().setStatusBarColor(Color.TRANSPARENT);
        }
        setContentView(R.layout.activity_weather);
        // 初始化各控件
……….
}
```

由于这个功能是 Android5.0 及以上的系统才支持，因此在代码中先做一个系统版本的判断，只有当版本号大于或等于 21，也就是 5.0 及以上时系统才会执行后面的代码。

接着调用 getWindow().getDecorView()方法拿到当前活动的 DecorView，再调用它的 setSystemUiVisibility（）方法来改变系统 UI 的显示，这里传入 View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN 和 View.SYSTEM_UI_FLAG_LAYOUT_STABLE 就表示活动的布局会显示在状态栏上面，最后调用一下 setStatusBarColor（）方法将状态栏设置成透明色。

可以实现让背景图和状态栏融合到一起了，但是天气界面的头布局几乎和系统状态栏紧贴到一起了。这是由于系统状态栏已经成为布局的一部分，因此没有单独为它留出空间，借助 Android:fitsSystemWindows 属性就可以解决，修改 activity_weather.xml 中的代码：

```
    <ScrollView
    android:id="@+id/weather_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="none"
    android:overScrollMode="never">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:fitsSystemWindows="true">
……….
```

运行程序。