Step 5: 重新设计显示天气的UI界面

目录

[目标： 1](#_Toc448425921)

[步骤： 2](#_Toc448425922)

[1、修改UI 2](#_Toc448425923)

[2、添加显示每天天气情况的layout文件 2](#_Toc448425924)

[3、为ListView生成用于显示的数据 3](#_Toc448425925)

[4、应用新UI 4](#_Toc448425926)

[5、优化UI，去掉不必要的文字 6](#_Toc448425927)

## 目标：

重新设计显示天气的UI界面，以显示各个元素，用LiveView显示今后几天的天气情况。

知识点：显示组件：ListView，为ListView提供数据的SimpleAdapter。

完成后的结果如下:



## 步骤：

### 1、修改UI

打开content\_main.xml，修改界面UI

1）删除TextVeiw，txtInfo

2）添加ListView，代码如下:

|  |
| --- |
| <**ListView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/listView"** /> |

### 2、添加显示每天天气情况的layout文件

在layout文件夹中创建layout文件: list\_item.xml，用于定义在ListView中每个Item的显示方式。

在list\_item.xml中添加4个TextView，分别把id设定为txtDate（显示日期）, txtTemperature（显示温度）, txtWeather(天气情况)，txtWind（风力）。

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="match\_parent"**>   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:textAppearance="?android:attr/textAppearanceMedium"  android:text="New Text"  android:id="@+id/txtDate"  android:layout\_alignParentTop="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentStart="true"** />   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="New Text"  android:id="@+id/txtTemperature"  android:layout\_alignParentTop="true"  android:layout\_alignParentRight="true"  android:layout\_alignParentEnd="true"** />   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="New Text"  android:id="@+id/txtWeather"  android:layout\_below="@+id/txtDate"  android:layout\_alignParentLeft="true"  android:layout\_alignParentStart="true"** />   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="New Text"  android:id="@+id/txtWind"  android:layout\_below="@+id/txtTemperature"  android:layout\_alignParentRight="true"  android:layout\_alignParentEnd="true"** /> </**RelativeLayout**> |

### 3、为ListView生成用于显示的数据

用SimpleAdapter为ListView提供数据，用法说明：

SimpleAdapter(Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to)

参数context：上下文，比如this。关联SimpleAdapter运行的视图上下文；

参数data：Map列表，列表要显示的数据，每条项目要与from中指定条目一致，即保存键值对；

参数resource：ListView单项布局文件的Id, 这个布局中必须包括了to中定义的控件id；

参数 from：一个被添加到Map上关联每一个项目列名称的列表，数组里面是列名称

参数 to：是一个int数组，数组里面的id是自定义布局中各个控件的id，需要与上面的from对应。

这个步骤就是为SimpleAdapter提供数据，把WeatherInfo对象转换成List<? extends Map<String, ?>>类型的对象。

编写WeatherInfo2SimpleAdapterData类完成这个步骤的工作，代码如下：

|  |
| --- |
| **public class** WeatherInfo2SimpleAdapterData {  **public static** List<Map<String, Object>> convert(WeatherInfo weatherInfo) {  List<Map<String, Object>> list = **new** ArrayList<>();  **if** (weatherInfo == **null** || weatherInfo.getData() == **null** || weatherInfo.getData().getForecast() == **null**) {  **return** list;  }  *// 获取每天的天气信息* List<Forecast> forecastList = weatherInfo.getData().getForecast();  *// 把Forecast转换成Map格式，添加到List<Map<String, Object>>中* **for** (Forecast f : forecastList) {  *// 创建每行的map对象* Map<String, Object> data = **new** HashMap<>();  *// 设置值* data.put(**"date"**, f.getDate());  data.put(**"temperature"**, f.getLow() + **"-"** + f.getHigh());  data.put(**"weather"**, f.getType());  data.put(**"wind"**, f.getFengli());  list.add(data);  }  **return** list;  } } |

### 4、应用新UI

修改MainActivity以应用新的UI。

1）编写方法showForecastList()显示数据，步骤如下：

* 把json字符串转换为WeatherInfo对象
* 判断是否正确的获取的数据，如果不是，显示错误信息，并返回
* 把WeatherInfo对象转换为List<Map<String, Object>>对象
* 创建SimpleAdapter对象
* 获取ListView并设置它的Adapter

代码如下：

|  |
| --- |
| **private void** showForecastList(String json) {  *// 把json字符串转换为WeatherInfo对象* Gson gson = **new** Gson();  WeatherInfo weatherInfo = gson.fromJson(json, WeatherInfo.**class**);   *// 判断是否正确的获取的数据，如果不是，显示错误信息，并返回* **if** (!weatherInfo.getDesc().equals(**"OK"**)) {  showError(**"获取天气信息出错。"**);  }  *// 把WeatherInfo对象转换为List<Map<String, Object>>对象* List<Map<String, Object>> dataList = WeatherInfo2SimpleAdapterData.*convert*(weatherInfo);   *// 创建SimpleAdapter对象* SimpleAdapter adapter = **new** SimpleAdapter(**this**, dataList,  R.layout.***list\_item***,  **new** String[]{**"date"**, **"temperature"**, **"weather"**, **"wind"**},  **new int**[]{R.id.***txtDate***, R.id.***txtTemperature***, R.id.***txtWeather***, R.id.***txtWind***}  );   *// 获取ListView并设置它的Adapter* ListView listView = (ListView) findViewById(R.id.***listView***);  listView.setAdapter(adapter); } |

说明：impleAdapter adapter = **new** SimpleAdapter(**this**, dataList,  
 R.layout.***list\_item***,  
 **new** String[]{**"date"**, **"temperature"**, **"weather"**, **"wind"**},  
 **new int**[]{R.id.***txtDate***, R.id.***txtTemperature***, R.id.***txtWeather***, R.id.***txtWind***} );

用**"date"**作为哈希表的key，从dataList中的每一行（或每一个对象）中取出对应的值，把它显示在R.id.***txtDate***指定的View上

2）修改NetTask类中的onPostExecute方法，用以显示新的UI

|  |
| --- |
| @Override **protected void** onPostExecute(String result) {  **if** (**exception** == **null**) {  *// 显示获取的信息  // TextView txtInfo = (TextView) findViewById(R.id.txtInfo);  // txtInfo.setText(result);* **showForecastList(result);**  } **else** {  *// 如果有异常，显示错误消息* showError(**"获取天气信息失败："** + **exception**.getMessage());  } } |

注意：被注释的部分是原来的显示方式。

**执行结果如下：**



### 5、优化UI，去掉不必要的文字

在上图的界面中“低温”“高温”是不必要的文字，去掉他们。

修改WeatherInfo2SimpleAdapterData类的convert方法，在准备数据时就去掉不必要的文字。

|  |
| --- |
| **public class** WeatherInfo2SimpleAdapterData {  **public static** List<Map<String, Object>> convert(WeatherInfo weatherInfo) {  …**for** (Forecast f : forecastList) {  …  ***// 温度区间，用正则表达式去掉不必要的文字* String t = f.getLow() + "-" + f.getHigh();  t = t.replaceAll("高温|低温|\\s\*", "");  data.put("temperature", t);**  …  }  **return** list;  } } |

结果如下：

