Step 8: 修改MainActivity，处理城市列表变化

目录

[目标 1](#_Toc448510093)

[步骤 1](#_Toc448510094)

[1、重构CityList 1](#_Toc448510095)

[2、使用CityList类：添加实例变量CityList cityList 3](#_Toc448510096)

[3、去掉输入城市的文本框，和相应按钮 3](#_Toc448510097)

[4、修改Toolbar的title为当前城市名。 4](#_Toc448510098)

[5、在Toolbar上添加“刷新”按钮，用于刷新当前城市的天气情况。 4](#_Toc448510099)

[6、处理CityListActivity返回 5](#_Toc448510100)

[7、MainActivity启动时立刻刷新天气情况 6](#_Toc448510101)

[8、Bug修复 7](#_Toc448510102)

## 目标

1）MainActivity类中增加两个实例变量，用于记录当前的城市名称，和城市在城市列表中的位置；

2）当CityListActivity返回时，根据需要更新当前的城市名称和位置。

## 步骤

### 1、重构CityList

1）增加两个实例变量，用于记录当前的城市名称，和城市在城市列表中的位置；

2）添加方法reload，获取新的城市信息，和更新当前城市

修改过的代码如下:

|  |
| --- |
| **public class** CityList {  **private** String **currentCityName** = **null**;  **private int currentCityIndex** = -1;     */\*\*  \* 从保存的数据中获取城市列表  \** ***@param activity*** *activity  \** ***@return*** *CityList  \*/* **public static** CityList load(Activity activity) {  *// 建立CityList对象* CityList list = **new** CityList();  *// 加载城市列表* list.reload(activity);  **return** list;  }   */\*\*  \* 当修改城市列表后调用此方法，获取新的城市信息，和更新当前城市  \** ***@param activity*** *activity  \*/* **public void** reload(Activity activity) {  SharedPreferences sharedPref = activity.getPreferences(Context.***MODE\_PRIVATE***);  String cityNames = sharedPref.getString(***StoreListKey***, **"[蒙自]"**);  *// 转换字符串到数组* Gson gson = **new** Gson();  String[] cityArray = gson.fromJson(cityNames, String[].**class**);  *// 赋值* Collections.*addAll*(**cityList**, cityArray);  *// 更新当前城市* **if** (**cityList**.isEmpty()) {  **currentCityIndex** = -1;  **currentCityName** = **null**;  **return**;  }  *// 确定是否需要更新当前城市* **if** (**currentCityName** == **null** || !**cityList**.contains(**currentCityName**)) {  **currentCityName** = **cityList**.get(0);  **currentCityIndex** = 0;  **return**;  }  *// 更新当前城市的索引位置* **currentCityIndex** = **cityList**.indexOf(**currentCityName**);  }  */\*\*  \* 取当前城市名  \** ***@return*** *当前城市名  \*/* **public** String getCurrentCityName() {  **return currentCityName**;  } } |

### 2、使用CityList类：添加实例变量CityList cityList

1）添加实例变量CityList cityList

2）修改loadCityName方法，加载citylist

### 3、去掉输入城市的文本框，和相应按钮

1）删除layout/content\_main.xml中相关的代码。完成后代码如下；

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  tools:context="cn.edu.uoh.cs.weatherforecast.MainActivity"  tools:showIn="@layout/activity\_main"**>   <**ListView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/listView"** /> </**LinearLayout**> |

2）删除相应的java代码

删除方法：

fetchWeatherInfo

saveCityName

loadCityName

### 4、修改Toolbar的title为当前城市名。

添加方法reloadCityList，在方法中获取当前城市，和修改title。

|  |
| --- |
| */\*\*  \* 重新加载城市名列表  \*/* **private void** reloadCityList() {  **if** (**cityList** == **null**) {  **cityList** = CityList.*load*(**this**);  } **else** {  **cityList**.reload(**this**);  }  Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  **if** (**cityList**.getCurrentCityName() != **null**) {  toolbar.setTitle(**cityList**.getCurrentCityName());  } **else** {  toolbar.setTitle(**"请添加城市"**);  } } |

### 5、在Toolbar上添加“刷新”按钮，用于刷新当前城市的天气情况。

1）添加刷新按钮，参见前面的操作

2）添加按钮按下的处理方法

|  |
| --- |
| */\*\*  \* 刷新天气情况  \*/* **private void** refreshWeatherInfo() {  *// 城市名称* String cityName = **cityList**.getCurrentCityName();  *// 调用异步任务类，访问网络，获取数据* NetTask task = **new** NetTask();  task.execute(cityName); } |

3）响应按钮操作，修改方法onOptionsItemSelected

|  |
| --- |
| @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  *// 刷新天气情况* **if** (id == R.id.***action\_refresh***) {  refreshWeatherInfo();  **return true**;  }  *// 设置城市列表* **if** (id == R.id.***action\_settings***) {  Intent intent = **new** Intent(**this**, CityListActivity.**class**);  startActivity(intent);  **return true**;  }  **return super**.onOptionsItemSelected(item); } |

### 6、处理CityListActivity返回

1）修改CityListActivity中的onOptionsItemSelected，使其发送返回event给MainActivity

|  |
| --- |
| @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  ***// 单击返回按钮时，调用finish方法，使主Activity能触发返回事件*****if (id == android.R.id.*home*) {  finish();  return true;  }**  *// 单击删除按钮时删除选中的城市* **if** (id == R.id.***remove\_city***) {  removeCity();  **return true**;  }  *// 单击添加按钮是显示添加UI* **if** (id == R.id.***add\_city***) {  View view = findViewById(R.id.***addLayout***);  view.setVisibility(View.***VISIBLE***);  **return true**;  }  **return super**.onOptionsItemSelected(item); } |

2) 修改启动CityListActivity的方式，由startActivity改为，startActivityForResult

|  |
| --- |
| **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  *// 刷新天气情况* **if** (id == R.id.***action\_refresh***) {  refreshWeatherInfo();  **return true**;  }  *// 设置城市列表* **if** (id == R.id.***action\_settings***) {  Intent intent = **new** Intent(**this**, CityListActivity.**class**);  **startActivityForResult(intent, 1);**  **return true**;  }  **return super**.onOptionsItemSelected(item); } |

3）添加响应返回事件的方法

|  |
| --- |
| */\*\*  \* 重CityListActivity返回后执行此方法：  \* 重新获取城市列表  \* 刷新天气情况  \** ***@param requestCode*** *\** ***@param resultCode*** *\** ***@param data*** *\*/* @Override **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  *// 重新获取城市列表* reloadCityList();  *// 刷新天气情况* refreshWeatherInfo(); } |

### 7、MainActivity启动时立刻刷新天气情况

1）修改CityListActivity中的onCreate方法

|  |
| --- |
| **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  setSupportActionBar(toolbar);  *// 恢复保存的城市名称* reloadCityList();  ***// 刷新天气* refreshWeatherInfo();** } |

### 8、Bug修复

测试时发现，MainActivity和CityListActivity中的城市列表是独立的，不能互相共享，经查询文档发现如下说明：

获取SharedPreferences的两种方式:

1 调用Context对象的getSharedPreferences()方法

2 调用Activity对象的getPreferences()方法

两种方式的区别:

调用Context对象的getSharedPreferences()方法获得的SharedPreferences对象可以被同一应用程序下的其他组件共享.

调用Activity对象的getPreferences()方法获得的SharedPreferences对象只能在该Activity中使用。

**修复bug1：**

而我们在CityList中调用的就是Activity对象的getPreferences()，所以数据不能共享，修改CityList，使用Context对象的getSharedPreferences()。

**修复bug2：**

再次load城市列表时，没有清除原来的城市，导致bug，

修改后的代码如下：

|  |
| --- |
| **public class** CityList {  */\*\*  \* 主key  \*/* **public final static** String ***Key*** = **"cn.edu.uoh.cs.weatherforecast"**;   */\*\*  \* 当修改城市列表后调用此方法，获取新的城市信息，和更新当前城市  \** ***@param context*** *context  \*/* **public void** reload(Context context) {  **SharedPreferences sharedPref = context.getSharedPreferences(*Key*, Context.*MODE\_PRIVATE*);**  …  *// 赋值* **cityList.clear();** Collections.*addAll*(**cityList**, cityArray);  }   */\*\*  \* 保存城市列表  \** ***@param context*** *\*/* **public void** save(Context context) {  …  *// 保存* **SharedPreferences sharedPref =context.getSharedPreferences(*Key*, Context.*MODE\_PRIVATE*);**  SharedPreferences.Editor editor = sharedPref.edit();  editor.putString(***StoreListKey***, cityJson);  editor.apply();  } |