**实验五 网络通信——天气预报**

学号：2021329600006 姓名：陈昊天 成绩：

班级：计算机科学与技术21（4）班 联系电话：13456982338

实验地点：10-414

实验日期：2023年11月29日

1. **实验目的**

掌握Android开发中网络请求和数据解析（以json为例）的常用方法。

1. **实验内容**
2. 可自选天气Web API接口。例如：

中国天气网接口：

http://www.weather.com.cn/data/sk/101010100.html

其中101110101是城市的代码，如果要查询其他城市的天气，只需要修改城市的代码即可，在中国天气网中城市代码如下：101010100=北京、101020100=上海、101210101=杭州、101280101=广州

百度天气接口：

https://api.map.baidu.com/weather/v1/?district\_id=330100&data\_type=all&ak=你申请的AK

1. 可选择和任意切换城市，例如北京、杭州、广州、上海的天气（阴晴雨雪），显示气温、湿度等信息。
2. 解析API接口获取的数据（json格式数据）。
3. 在app中展现查询到的天气结果。

**三、实验过程(实验的设计思路、关键源代码、成果截图等)**

**实验的设计思路**

1. 选择API接口：选择OpenWeatherMap作为提供天气信息的Web API。

2. 界面设计：设计一个简洁的用户界面来展示天气信息。使用一个下拉菜单来选择城市，一个文本视图来显示请求结果，以及其他文本视图来展示天气描述、温度、湿度等信息。

3. 编写网络请求代码：使用OkHttp库来发起网络请求，获取选定城市的天气数据。网络请求需要在非UI线程上执行，以避免阻塞主线程。

4. 解析JSON数据：一旦收到API响应，解析JSON数据以获取所需的天气信息。使用JSONObject类及相关方法来提取天气数据。

5. 更新UI：解析完JSON数据后，需要将解析出的数据更新到UI上。这应该在主线程中完成。可以使用runOnUiThread()方法来实现。

6. 异常处理：在网络请求和JSON解析过程中，可能会出现异常。需要妥善处理这些异常，并给用户合适的提示。

7. 测试：测试应用是否能够正确发送请求、解析数据并更新界面，确保在不同的城市选择下都能正常工作。

**关键源代码**

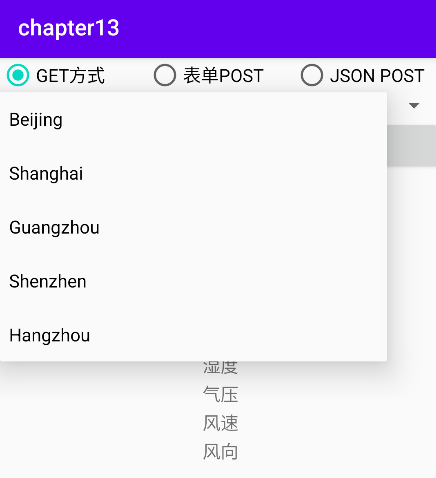
设置Activity的布局、初始化UI组件、设置下拉菜单、设置单选按钮组的监听器、设置按钮点击监听器

@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_okhttp\_call***);  
 **ll\_login** = findViewById(R.id.***ll\_login***);  
 **et\_username** = findViewById(R.id.***et\_username***);  
 **et\_password** = findViewById(R.id.***et\_password***);  
 **tv\_result** = findViewById(R.id.***tv\_result***);  
 RadioGroup rg\_method = findViewById(R.id.***rg\_method***);  
  
 Spinner spinnerCity = findViewById(R.id.***spinner\_city***);  
 ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(**this**,  
 R.array.***city\_list***, android.R.layout.***simple\_spinner\_item***);  
 adapter.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
 spinnerCity.setAdapter(adapter);  
  
 rg\_method.setOnCheckedChangeListener((group, checkedId) -> {  
 **mCheckedId** = checkedId;  
 **int** visibility = **mCheckedId** == R.id.***rb\_get*** ? View.***GONE*** : View.***VISIBLE***;  
 **ll\_login**.setVisibility(visibility);  
 });  
 findViewById(R.id.***btn\_send***).setOnClickListener(v -> {  
 **if** (**mCheckedId** == R.id.***rb\_get***) {doGetWeather();  
 }  
 });  
}

发起一个网络GET请求来获取指定城市的天气信息，并将返回的数据展示在用户界面上。包括创建OkHttpClient实例、获取选中的城市、构建请求URL、构建Request对象、异步发起网络请求、处理失败响应、处理成功响应、解析JSON数据、更新UI、异常处理

**private void** doGetWeather() {  
 OkHttpClient client = **new** OkHttpClient();  
 Spinner spinnerCity = findViewById(R.id.***spinner\_city***);  
 String selectedCity = spinnerCity.getSelectedItem().toString();  
String url = **"https://api.openweathermap.org/data/2.5/weather?q="** + selectedCity + **"&appid=01b2fd1004eb0888ea275b0a08d93b7a&units=metric&lang=zh\_cn"**;  
 Request request = **new** Request.Builder()  
 .url(url)  
 .build();  
  
 client.newCall(request).enqueue(**new** Callback() {  
 @Override  
 **public void** onFailure(Call call, IOException e) {  
 runOnUiThread(() -> **tv\_result**.setText(**"请求天气预报接口出错："** + e.getMessage()));  
 }  
  
 @Override  
 **public void** onResponse(Call call, Response response) **throws** IOException {  
 String resp = response.body().string();  
 **try** {  
 JSONObject jsonObject = **new** JSONObject(resp);  
 String cityName = jsonObject.getString(**"name"**);  
 JSONObject weatherObj = jsonObject.getJSONArray(**"weather"**).getJSONObject(0);  
 String weatherDescription = weatherObj.getString(**"description"**);  
 JSONObject mainObj = jsonObject.getJSONObject(**"main"**);  
 **double** temp = mainObj.getDouble(**"temp"**);  
 **double** feelsLike = mainObj.getDouble(**"feels\_like"**);  
 **double** humidity = mainObj.getDouble(**"humidity"**);  
 **double** pressure = mainObj.getDouble(**"pressure"**);  
 JSONObject windObj = jsonObject.getJSONObject(**"wind"**);  
 **double** windSpeed = windObj.getDouble(**"speed"**);  
 **double** windDirection = windObj.getDouble(**"deg"**);  
runOnUiThread(() -> {  
 ((TextView) findViewById(R.id.***tv\_city\_name***)).setText(cityName);  
 ((TextView) findViewById(R.id.***tv\_weather\_description***)).setText(**"天气："** + weatherDescription);  
 ((TextView) findViewById(R.id.***tv\_temperature***)).setText(String.*format*(**"温度：%.1f°C"**, temp));  
 ((TextView) findViewById(R.id.***tv\_feels\_like***)).setText(String.*format*(**"体感温度：%.1f°C"**, feelsLike));  
 ((TextView) findViewById(R.id.***tv\_humidity***)).setText(String.*format*(**"湿度：%d%%"**, (**int**) humidity));  
 ((TextView) findViewById(R.id.***tv\_pressure***)).setText(String.*format*(**"气压：%d hPa"**, (**int**) pressure));  
 ((TextView) findViewById(R.id.***tv\_wind\_speed***)).setText(String.*format*(**"风速：%.1f m/s"**, windSpeed));  
 ((TextView) findViewById(R.id.***tv\_wind\_direction***)).setText(String.*format*(**"风向：%d°"**, (**int**) windDirection));  
 });  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 runOnUiThread(() -> **tv\_result**.setText(**"解析天气数据出错："** + e.getMessage()));  
 }  
 }  
 });  
}

**成果截图**



**四、实验心得**

通过这次实验，我深入体会到了Android开发中网络请求和JSON数据处理的重要性。在实验过程中，我首先选择了合适的天气Web API，并学会了如何使用OkHttp库来发起网络请求。这个库的使用相对直观，让我能够高效地发送HTTP请求并处理异步响应。更为重要的是，我理解了在Android中进行网络操作时，必须在非UI线程上执行网络请求以避免阻塞主线程，这对于保证应用的流畅性至关重要。

解析JSON数据是我在这次实验中学到的另一个关键技能。通过实际操作，我掌握了如何使用JSONObject和JSONArray等类来提取和处理JSON格式的数据。我意识到准确地提取所需信息并正确地处理可能出现的异常是确保用户体验的关键。

这次实验也强化了我的UI设计思维。我学会了如何动态地更新UI组件，如文本视图来显示天气信息，以及如何使用Spinner来提供一个城市选择的下拉列表。通过实验，我明白了用户界面在应用程序中的重要性，以及如何使它既美观又实用。

这次实验也提醒了我测试的重要性。在开发过程中，我不断尝试不同的城市，确保应用能够正确处理各种情况下的数据，并且在遇到问题时及时调整代码。这种实践使我更加注重细节，也提高了解决问题的能力。

通过这次实验，我不仅掌握了Android网络编程的关键技巧，而且也在实际应用中提升了我的编程能力和问题解决能力。这将为我之后的学习和开发工作奠定坚实的基础。