Android多线程与网络请求

多线程

先吹一吹概念，啥是多线程？下面是一段毫无意义的百度百科讲解

> 多线程（multithreading），是指从软件或者硬件上实现多个线程并发执行的技术。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。具有这种能力的系统包括对称多处理机、多核心处理器以及芯片级多处理或同时多线程处理器。在一个程序中，这些独立运行的程序片段叫作“线程”（Thread），利用它编程的概念就叫作“多线程处理”。

接下来用人话讲解一下啥是多线程，举个例子

> 张涛学姐每天要处理大量的问题，今天是一个快乐的周六，张涛学姐打开自己的待办事项，看了一下，自己需要处理以下几件事情：
>
> - 回答学妹A的问题
> - 找张煜收他的作业
> - 去GYM健身
>
> 按照正常人的思路，涛姐姐会一件一件的来做，比如先回答学妹的问题，再去收张煜的作业，最后去健身。但涛姐姐可非等闲之辈，她可以同时完成多项任务，比如在去健身房的路上一边打电话给张煜要作业一边打字回答学妹问题，这时我们可以称张涛学姐处于多线程状态

总结一下，涛涛学姐这种一个人同时干多件事情的状态，可以称之为多线程，在实际的开发情境中，更有可能出现的情景是我的程序在同时进行着网络请求和前台交互，举一个例子

> 你点开哔哩哔哩，然后下拉刷新，这时候App需要进行网络请求，也就是去获得一批刷新之后的视频，众所周知，网络请求是需要时间的，如果不采取多线程的思路，在进行网络请求的时候就啥事情都做不了，就会造成App界面的卡死，而如果可以进行多线程，一边进行网络请求，一边维持着App页面的正常刷新，就可以实现在网络请求的时候不引起App页面卡死

上面举了一个不引起App UI界面卡死的例子，事实上系统也是不允许在主线程进行网络请求的，事实上不仅是网络请求，任何高耗时的操作都不被允许在主线程进行，UI线程的卡死将会引起界面卡顿，在用户交互上，这是不被允许的，谁都不想看到自己的App卡在那里，一动不动

接下来复习一些线程的相关语法，下面是一系列简单的开新线程的写法

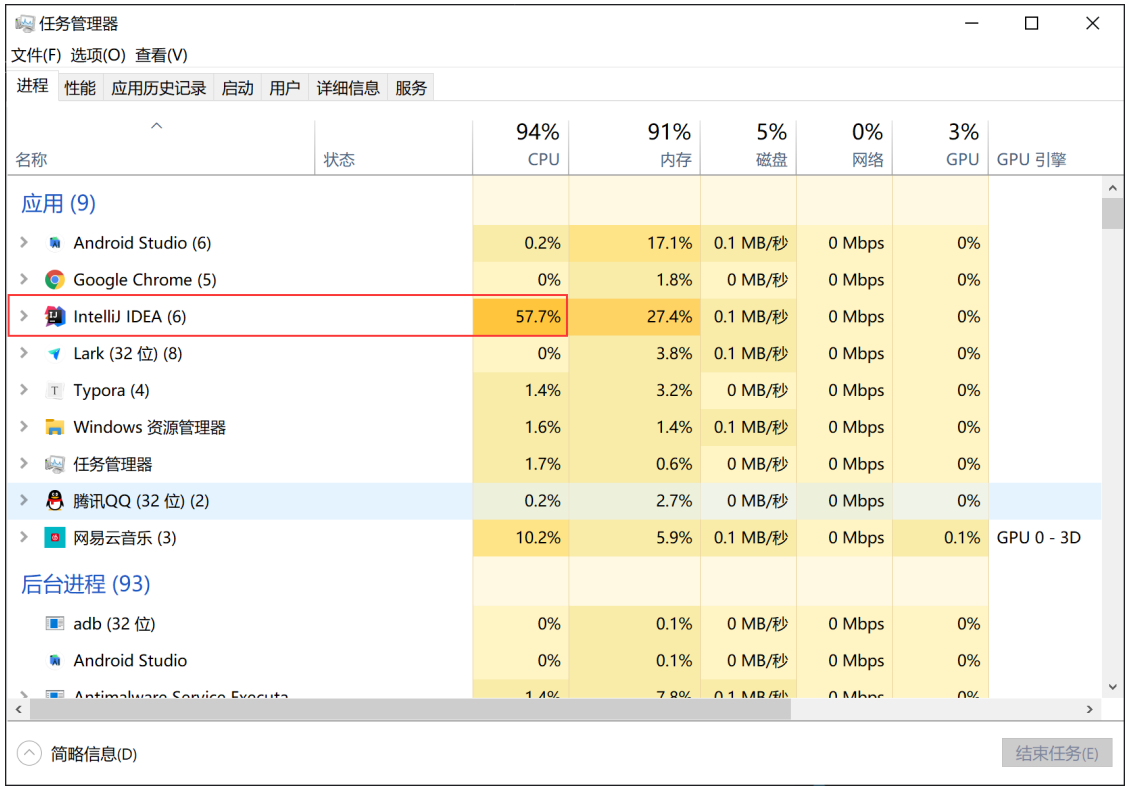
```
public class Main {
    public static void main(String[] args) {

        //new一个线程
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                //在这里写下你想要执行的代码
                Main.showCurrentThread();
            }
        };
        Thread thread = new Thread(runnable);
        thread.start();
    }

    static void showCurrentThread(){
        System.out.println("这里是" + Thread.currentThread() + "线程");
```

```
    }
}
```

这里新开一个线程类似于新增一个任务，而程序就相当于执行任务的人，任务的执行者在同时执行多个任务，是不是很像上文中的涛哥

> Tip1：不知道大家有没有想过，早期的单核CPU是如何实现多线程的？其实单核CPU实现多线程的方式是快速的在多个线程之间调度，先运行一些线程1的代码，然后再跳转调度去运行另一个线程的代码，由于跳转的相当快，使用者站在使用的角度就会感觉这些事情是同时进行的
>
> Tip2：冬季电脑变暖手宝秘技，死循环new线程，下面这几行代码，将会让CPU负荷直接拉满

```
public class Main {
 public static void main(String[] args) {
     while(true){
         new Thread().start();
     }
 }
}
```



JSON格式

现在仅仅靠本地功能取胜的App越来越少，绝大多数App都会和网路和数据挂钩，那么App网络请求会获得什么呢？我们又如何实现网络请求呢？

JSON格式与信息传递

首先我们要认识到，网络请求的本质是一种通信，即后端（服务器）和前端（移动终端）的一种通信，根据我所剩不多的信息论知识，通信的基本模型分为三部分，信源信道和信宿，信源讲信息编码之后通过信道传输给信宿，信宿再根据规则进行解码，获得信源想要发送的信息

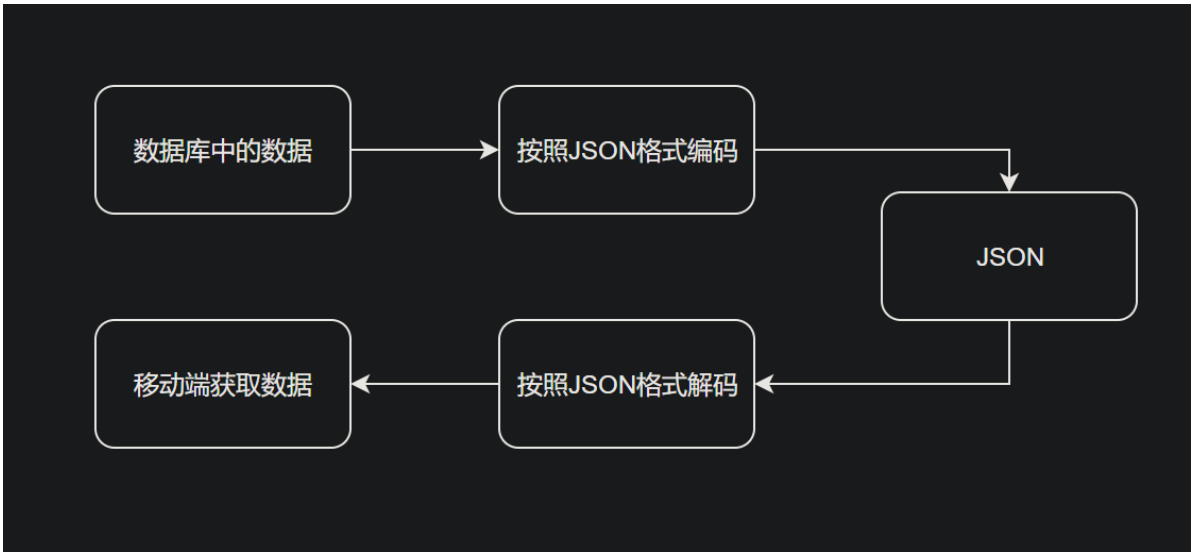
```
信息------->编码------->信道------->解码------->信息
```

JSON（**J**ava**S**cript **O**bject **N**otation，JavaScript对象表示法）是一种由道格拉斯·克罗克福特构想和设计、轻量级的资料交换语言，该语言以易于让人阅读的文字为基础，用来传输由属性值或者序列性的值组成的数据对象

虽然说网络请求中真正的编码解码并不是JSON，而是底层的一些编码规则，但是我们也可以近似的将JSON看成是一种编码格式，一种信息传输的规范，这里举一个例子。

> 期末的大学物理考试，张煜即将面临挂科，这时的他像坐旁边的潼哥求救，潼哥甩了一张纸条给了张煜，上面是这次大学物理考试的选择题答案，那么要怎么才能保证潼哥想传递出去的信息和张煜接收到的信息是相同的呢？他们俩就要事先约定好一种格式，潼哥依据这种格式写纸条，张煜按照这种格式解析纸条的内容，就可以完成信息的传递

JSON就是后端服务器和前端约定的一种**格式**，后端依据这种格式，将数据库中的数据编写成JSON字段，这些字段传输到前端终端的时候，终端按照这种格式的规定解析这些字段，最终达到实现信息传递的目的



接下来我们具体的看一下JSON的具体格式究竟是怎样的

基本数据类型和String的JSON转换格式

JSON的数据传输思想是键值对存储和读取，也就是一个“键”对应一个“值”，举上面的大物考试的例子，我我们可以认为选择题答案分为两个部分，一个是题号，一个是答案，这样的话就需要两个键值对，即**题号（键）**对应一个**表示题号的数字（值）**，以及**答案（健）**对应**表示答案的一个字符串（值）**，用java的思想来说，就是如果要想存储一道选择题的答案，你需要编写一个这样的数据类

```
public class AnswerData {
    public int number;
    public String answer;
}
```

假如说其中一个答案的题号是2，答案是a，这样的一个数据用JSON表示就是：

```
{ "number":2 , "answer":"a" }
```

下面来详细了解一下JSON的语法规则

JSON以"**{**"标志开始，以"**}**"标志结束，整体是采取了键值对的方式进行存储，具体格式为

```
"参数的名称（键）":"参数的值（值）"
```

或者用java中类的属性去对照理解

```
"属性名称":"属性的值"
```

不同的键值对之间用逗号隔开

数组的JSON转换格式

JSON的数组存储结构为

```
"数组名称": [
    数组元素1,
    数组元素2,
    数组元素3,
    数组元素4,
    ...
]
```

即数组名称（键）后面跟上一个方括号，方括号中不同数组元素用逗号隔开，**注意**，这里的数组元素不一定是基本数据类型，也可能是基本数据类型的组合体，甚至可以是基本数据类型和数组的组合体，这里先简单的举出一个例子，有关更复杂的情况稍后将会提到，下面是一个由年份组成的数组

```
"years": [
    2018,
    2019,
    2020
]
```

对象(符合数据类型)的JSON对应格式

就像是我们自己写java代码一样，我们的类的属性不止包含基本数据类型，同时也包含着其他的类，其他的类中又有其他的数据类型，这里还是举一个简单的j例子：如果要构建一个学校类，学校类需要具备以下属性：一个名字（String类型），一个学生数组（Student类型的数组），一个校长（President类型），然后我们假设学生类有两个属性：姓名（String）和年龄（int），校长有三个属性：姓名（String），年龄（int），工资（int），那么这个学校的JSON又是一个怎样的格式呢，我直接丢出来给大家看看

```
{
    "school": {
        "name": "某中学",
        "presidnet":{
            "name": "WXZ",
            "salary": 100000000,
            "age": 30
        },
        "student":[
            {
                "name": "A",
                "age":19
            },
            {
                "name": "B",
                "age":19
            },
            {
```

```
                "name": "C",
                "age":18
            }
        ]
    }
}
```

这就是JSON的大体规则，我们解析JSON数据同样的也要用到这些规则

那么怎么进行JSON的解析呢?

JSON的解析

我们手动解析JSON的方案是使用JSONObject类，本质上的思路还是按照JSON的数据格式，根据键获取对应的值，不论这个值是基本数据类型，还是复合数据类型

就拿刚刚的那个学校的JSON来举一个例子吧

```
private void jsonDecodeTest(){
        String jsonData = "{\"school\":{\"name\":\"某中学\",\"presidnet\":
{\"name\":\"WXZ\",\"salary\":100000000,\"age\":30},\"student\":
[{\"name\":\"A\",\"age\":19},{\"name\":\"B\",\"age\":19},
{\"name\":\"C\",\"age\":18}]}}";
        try {
            JSONObject jsonObject = new JSONObject(jsonData);
            JSONObject jsonObjectSchool = jsonObject.getJSONObject("school");
            String schoolName = jsonObjectSchool.getString("name");
            Toast.makeText(this, schoolName, Toast.LENGTH_LONG).show();

            JSONArray jsonArrayStudents =
jsonObjectSchool.getJSONArray("student");
            for (int i = 0; i < jsonArrayStudents.length(); i++) {
                JSONObject jsonObjectStu = jsonArrayStudents.getJSONObject(i);
                String stuNum = jsonObjectStu.getString("name");
                Toast.makeText(this, stuNum, Toast.LENGTH_SHORT).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

这个过程就像剥洋葱一样，一层一层的来，一层一层的解析

网络请求

接下来详细的看一下如何在Android中进行网络请求

网络权限申请

Android中涉及到一些权限的申请，像是诸如网络请求这种行为需要像系统申请权限，[这个是谷歌官方的Android权限概览]，将来在开发中很有可能还会遇到其他权限的申请，届时可以在这个连接中找到对应的标签名称，下面是申请网络权限的方法

首先是在AndroidManinfest里添加`<uses-permission android:name="android.permission.INTERNET"/>`标签，具体的添加位置如下

```
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">
    <!--在这里添加权限申请-->
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

添加了这一行申请，我们就可以轻松的访问任何以`https`为协议的网络请求了，但是在Android 9及以上的版本，对于`http`格式的网络请求，我们还需要额外的写一个xml文件进行一个请求安全的适配，详细信息可以参考Google的这篇文档

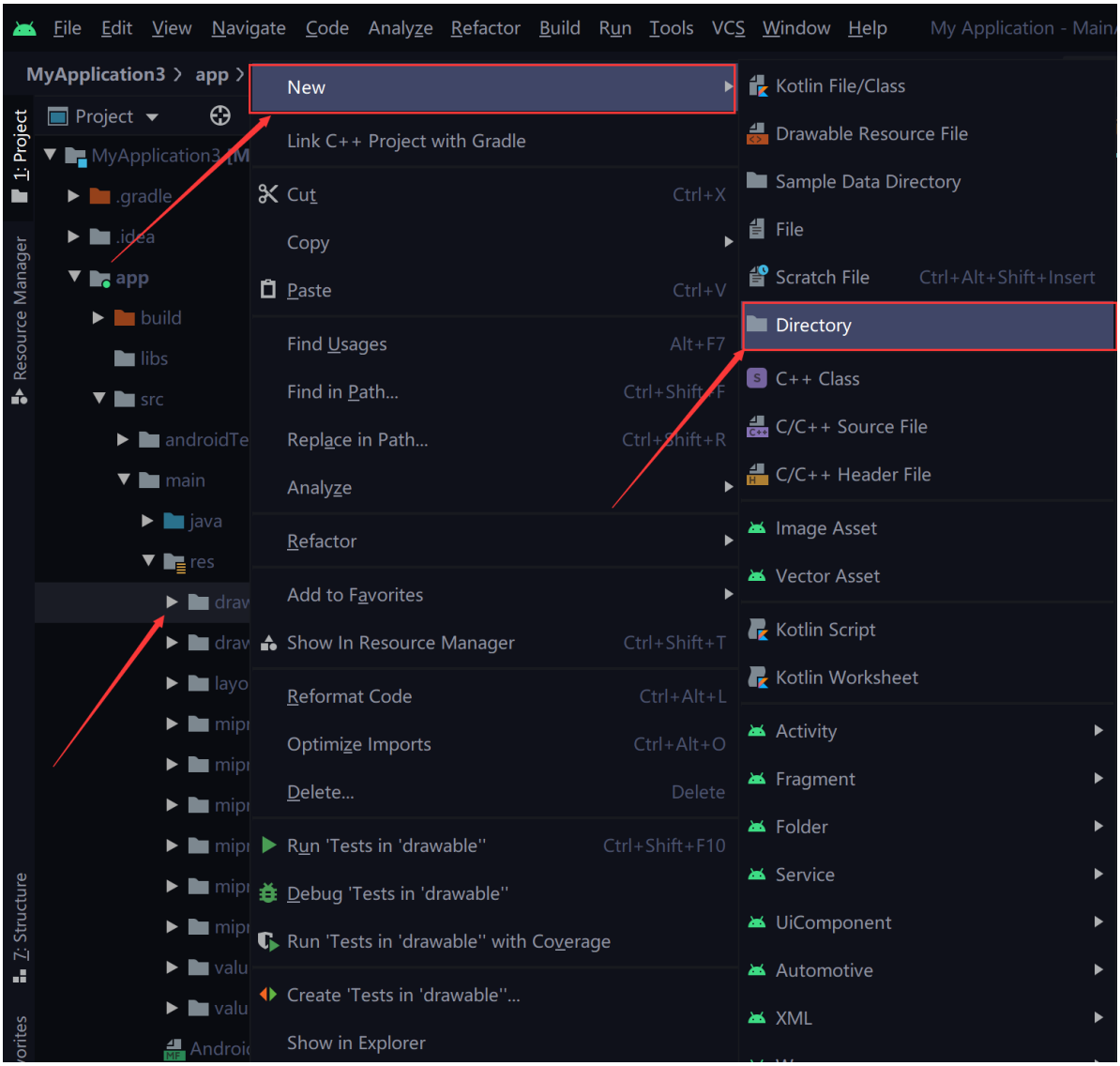
一般而言我们只需要在项目的res目录下新建一个名叫xml的文件夹，然后在其中写一个名为`net_config`(名字可以随便取)的xml文件（如果不确定如何建立文件夹和新建xml文件可以参考后面的附图），在其中写入下面的这些内容

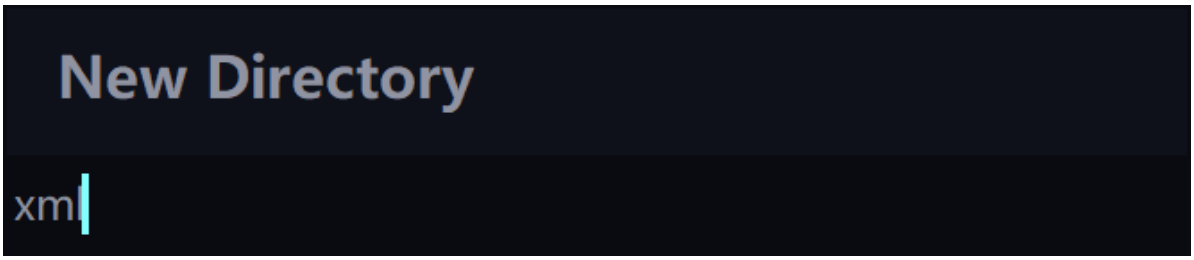
```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true"/>
</network-security-config>
```

之后再AndroidManifest添加这么一行就可以了

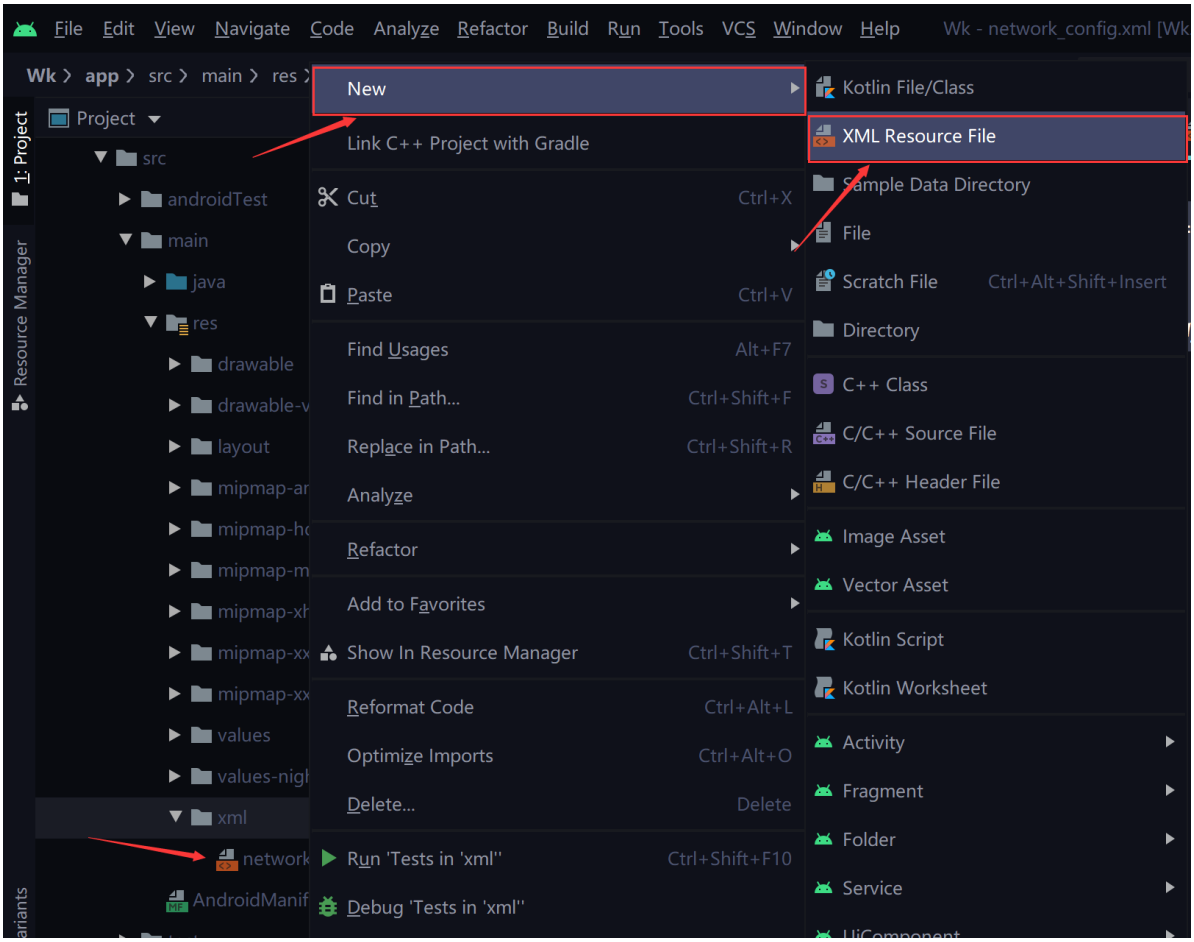
```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:networkSecurityConfig="@xml/network_config"      <---这一行（绿色部分仅仅起指示作用，不要写在manifest里面）
    android:supportsRtl="true"
    android:theme="@style/Theme.Wk"
    tools:targetApi="n">
```

下为新建名为xml的文件夹的方式





之后新建一个名为`net_config`的xml文件



之后切换到code，吧上面的东西复制进去就可以了

使用HttpUrlConnection进行网络请求

虽然说目前市面上有相当多优秀的网络请求库，但是目前我们还是要使用最原生的网络请求工具HttpUrlConnection

网络请求的方式

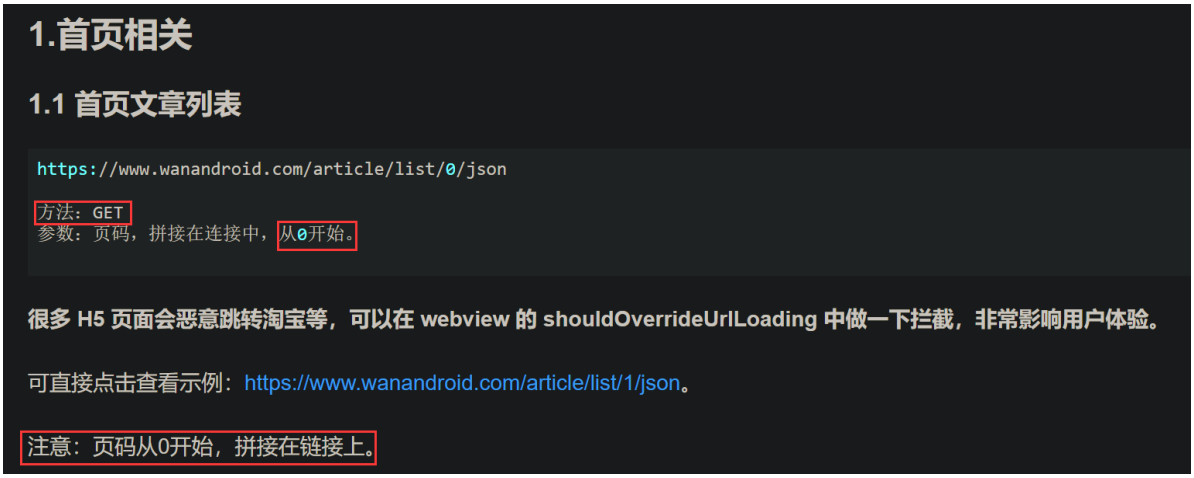
其实种类相当多，这里仅仅讲一下最常用的POST和GET请求，足够大家在将来的一段时间内使用

先教大家咋看接口文档，以及咋简单的使用postman工具

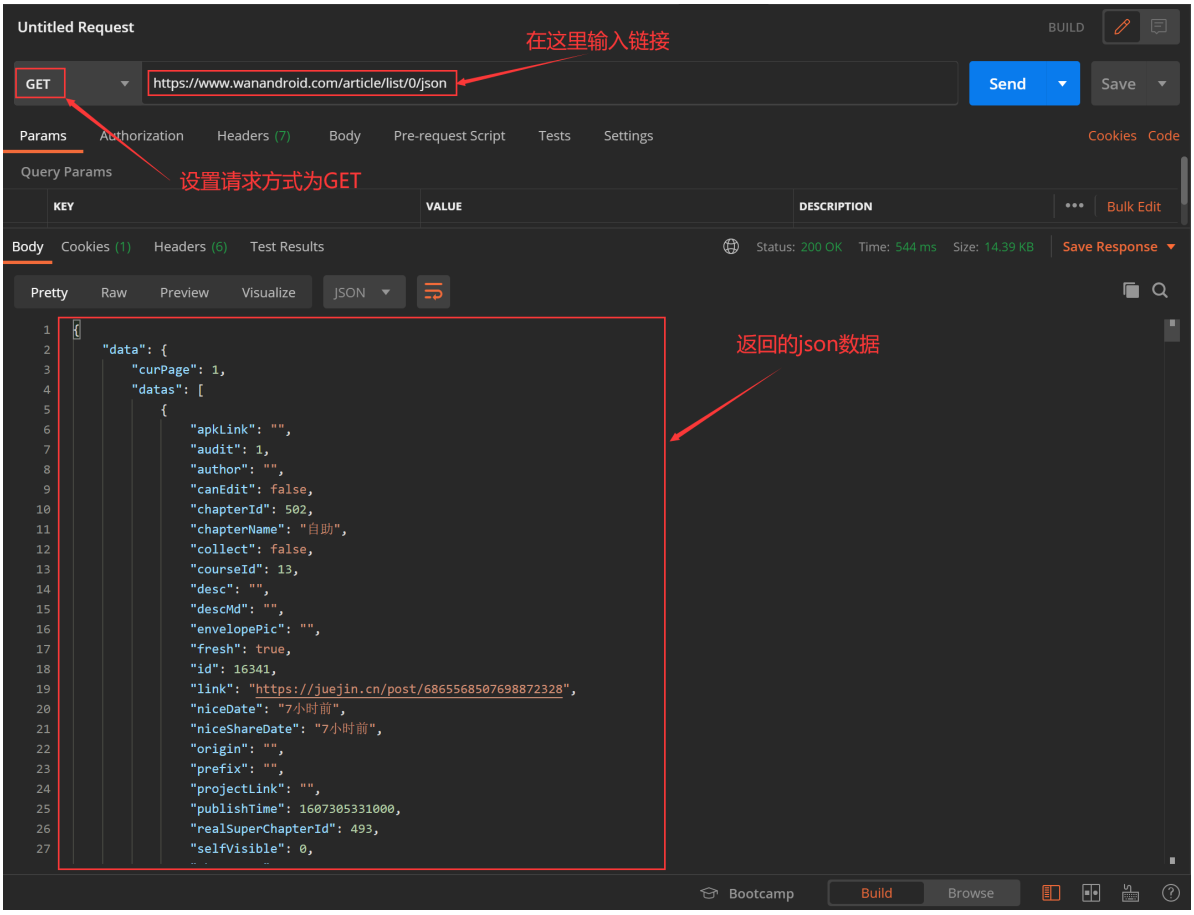
先甩一个postman的下载链接，几乎是全程无脑安装，不再赘述，这个是32位，这个是64位

如果不懂得如何看接口文档的话，可能真的会无从下手，所以先说一说咋看接口文档

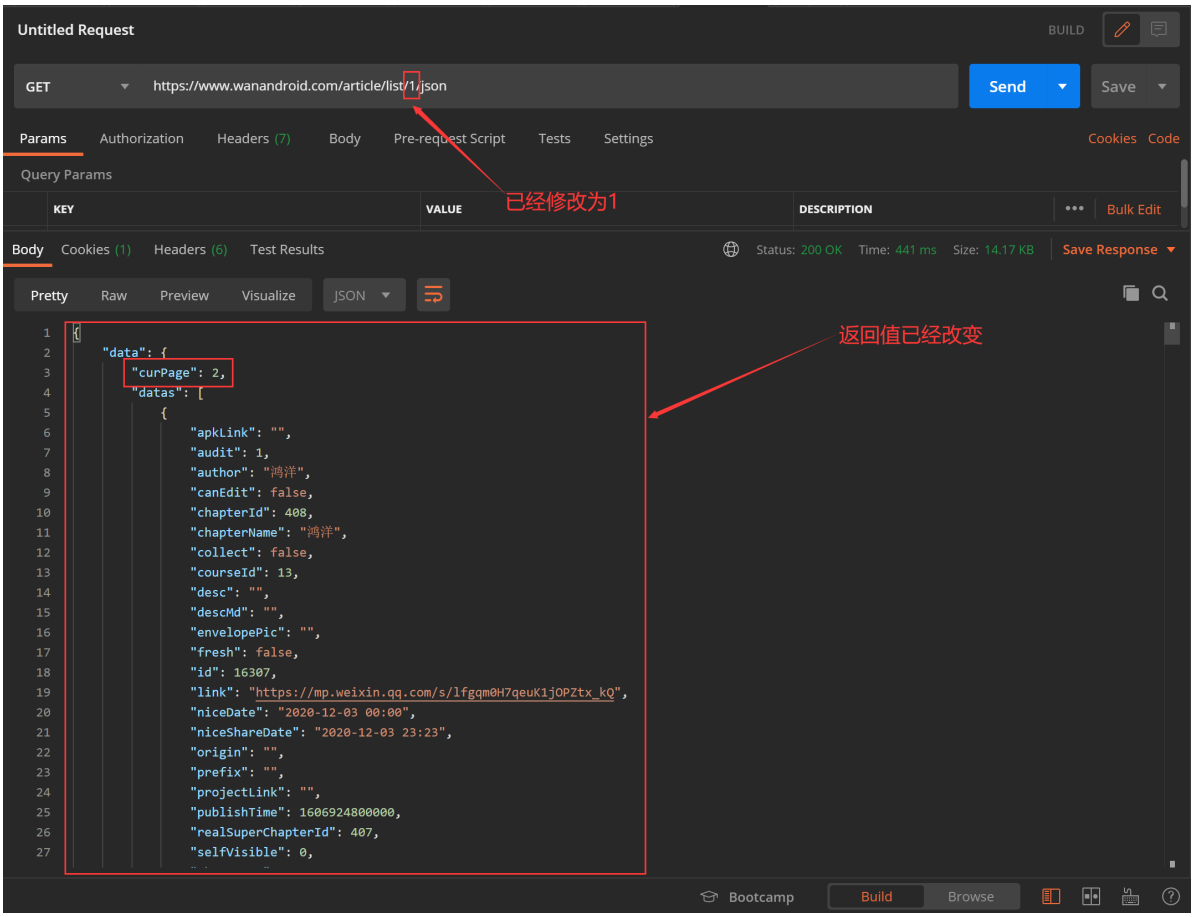
下面是一个简单的接口说明，看看它说明了那些信息



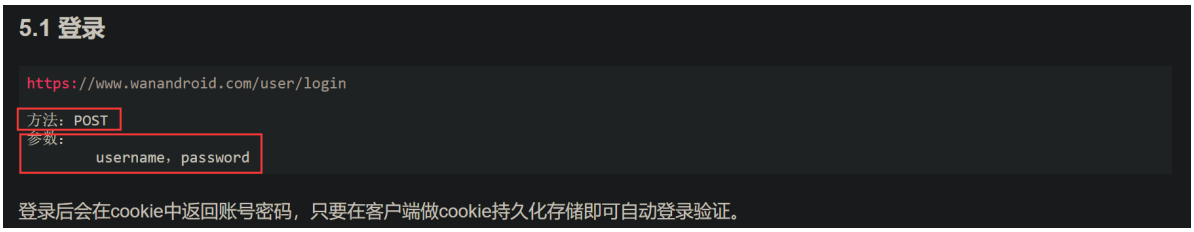
首先，方法为GET，表示这个接口需要通过GET的方式进行请求（GET与POST将会在后面进行具体说明），再次，参数为页码，要求拼接在链接中，也就是说可以通过修改`list/0/json`中间的那个数字来改变获取的首页文章的页数，我们用postman测试一下



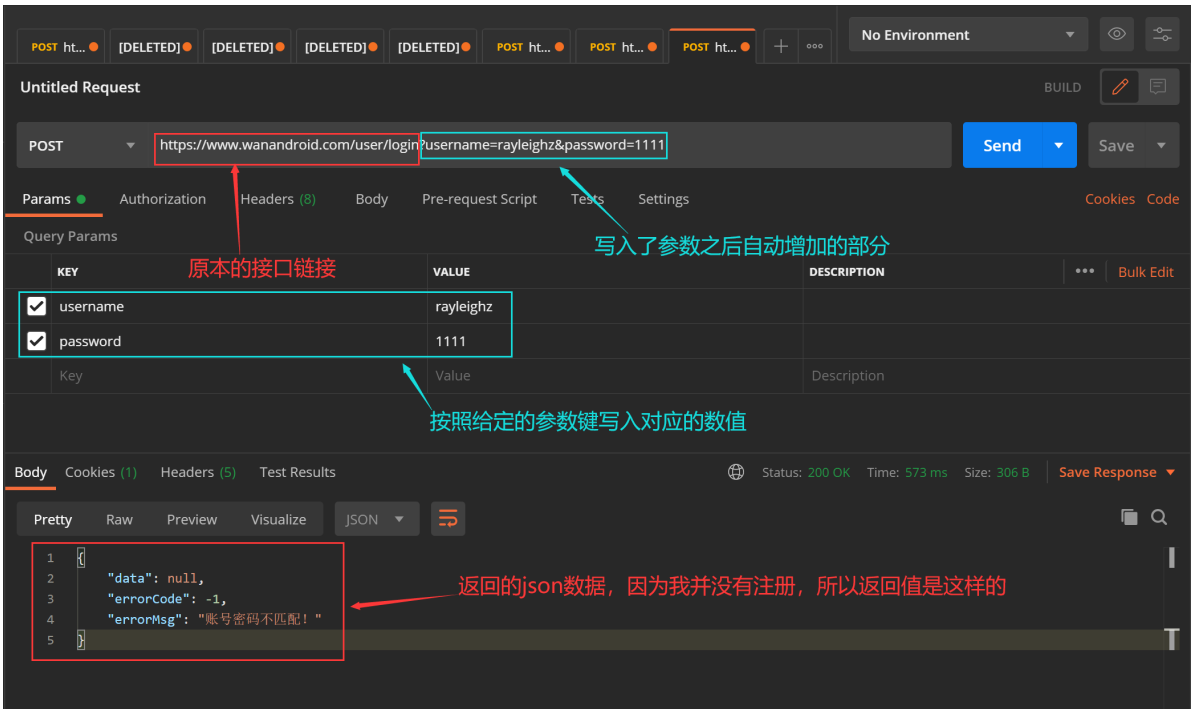
然后修改`list/0/json`中间的0为1，再来看看返回的结果



接下来再看看另一个接口



这里要求请求格式为POST，需要传递两个参数进去，两个参数的键分别为`username`，`password`，如果我们用postman来测试这个接口的话，要这样操作



GET请求

GET请求一般用于从服务器中获取数Z据，其参数与键值对是直接暴露在url之中的，相对而言比较不安全，所以一般不用GET请求传递参数，通过HttpUrlConnection发送简单的GET请求的方式并接收返回值的方式如下

```
private void sendGetNetRequest(String mUrl) {
        //开启子线程执行网络请求
        new Thread(
                () -> {
                    try {
                        URL url = new URL(mUrl);//这一行可能会导致
MalformedURLException(url格式异常)，所以要try catch掉
                        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                        connection.setRequestMethod("GET");//设置请求方式为GET
                        connection.setConnectTimeout(8000);//设置最大连接时间，单位为
毫秒，超出这个设定的时间将会导致连接超时
                        connection.setReadTimeout(8000);//设置最大的读取时间，单位为
毫秒，超出这个设定的时间将会导致读取超时
                        connection.connect();//正式连接
                        InputStream in = connection.getInputStream();//从接口处获取
输入流
                        String responseData = StreamToString(in);//这里就是服务器返
回的数据
                    } catch (Exception e) {
```

```
                    e.printStackTrace();
                }
            }
        ).start();
    }

    //将输入流转化为字符串的函数
private String StreamToString(InputStream in) {
        StringBuilder sb = new StringBuilder();//新建一个StringBuilder，用于一点一点构架字符串
        String oneLine;//流转换为字符串的一行
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));//由inputStream获取BufferReader，用于读取二进制流
        try {
            while ((oneLine = reader.readLine()) != null) {//readLine方法将读取一行数据，并转化为String类型，当读取的一行数据为null时，就代表当前数据已经读取完毕
                sb.append(oneLine).append('\n');//拼接字符串并且增加换行，提高可读性
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                in.close();//关闭InputStream
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();//将拼接好的字符串返回出去
}
```

POST请求

post请求和get请求在实现上大同小异，这里丢一个例子给大家看看，可以重点看一下如何实现在post请求中传递参数

```
private void sendPostNetRequest(String mUrl, HashMap<String, String> params) {
    //开启子线程执行网络请求
    new Thread(
            () -> {
                try {
                    URL url = new URL(mUrl);//这一行可能会导致MalformedURLException(url格式异常)，所以要try catch掉
                    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                    connection.setRequestMethod("POST");//设置请求方式为POST
                    connection.setConnectTimeout(8000);//设置最大连接时间，单位为毫秒，超出这个设定的时间将会导致连接超时
                    connection.setReadTimeout(8000);//设置最大的读取时间，单位为毫秒，超出这个设定的时间将会导致读取超时
                    connection.setDoOutput(true);//允许输入流
                    connection.setDoInput(true);//允许输出流
                    StringBuilder dataToWrite = new StringBuilder();//构建参数值
                    for (String key : params.keySet()) {

 dataToWrite.append(key).append("=").append(params.get(key)).append("&");//拼接参数
                    }
```

```
                    connection.connect();//正式连接
                    OutputStream outputStream = connection.getOutputStream();//开
启输入流
                    outputStream.write(dataToWrite.substring(0,
dataToWrite.length() - 1).getBytes());//去除最后一个&
                    InputStream in = connection.getInputStream();//从接口处获取输入
流
                    String responseData = StreamToString(in);//这里就是服务器返回的
数据
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
    ).start();
}
```

在主函数中要这么调用

```
HashMap<String, String> map = new HashMap<>();
map.put("username", "你的用户名");
map.put("password", "用户的密码");
sendPostNetRequest("https://www.wanandroid.com/user/login", map);
```

将参数值转化为HashMap进行存储，并且写入到网络连接中

跨线程通信

在Android中，UI操作必须要在UI线程（主线程中执行），而网络请求获取到的数据往往是在子线程中，比如我现在已经在子线程中通过网络请求获取到了所需要的文字，现在想要展示在自己的TextView上，要怎么做呢?

肯定不能在子线程中直接给TextView setText，因为正如前文所说，Android只允许在主线程中更新UI，那么要想在网络请求之后刷新视图，就需要通过一种方式将请求获得的数据发送给主线程，而实现这种跨线程的接口回调的方法就是**Handler**

Handler的简单使用

Handler背后的消息机制其实相当有趣，也相当重要，有兴趣的同学可以去了解一下，这里只简单介绍一下Handler的使用

创建一个内部类类继承自Handler，并重写其handleMessage方法

```
private class MHandler extends Handler{
        @Override
        public void handleMessage(@NonNull Message msg) {
            super.handleMessage(msg);
            //在这里写下你对发送过来的message的处理
        }
}
```

在主线程中new一个MHandler，并让他在子线程中发送（sendMessage）方法

首先是构造Message，Message顾名思义，就是在子线程和主线程之间通信的信息，比如网络请求获得的数据，或者一些其他的东西，都可以用Message实现在子线程和主线程之间的传递。这里用传递网络请求获得的字符串来举一个例子

```
String responseData = StreamToString(in);//这里就是服务器返回的数据
Message message = new Message();//新建一个Message
message.obj = responseData;//将数据赋值给message的obj属性
mHandler.sendMessage(message);//通过在主线程中实例化好的mHandler发送出Message
```

而这个发送出去的message就会在MHandler中重写的handleMessage方法中得到处理

```
private class MHandler extends Handler{
        @Override
        public void handleMessage(@NonNull Message msg) {
            super.handleMessage(msg);
            String responseData = msg.obj.toString();//这里的数据就是发送时赋值给obj
的那个String
            //现在就已经回到主线程之中了，你可以对这个数据进行任何处理，包括更新UI
        }
}
```

这样就完成了从子线程到主线程的跨线程通信

关于Warning

这样去新建一个名为MHandler的内部类系统会提示：`Inner class 'MHandler' may be 'static'. Default constructor in android.os.Handler is deprecated. This` Handler` class should be static or leaks might occur (com.example.wk.MainActivity.MHandler). 简而言之就是会导致内存泄漏，学有余力的同学可以去学习一下有关Java的垃圾回收机制以及强软弱虚四种不同引用的相关知识，并思考如何处理这个Warning，当然，如果暂时无法理解这些可以放任不管，在经过了下学期的学习后这个就不再是问题。