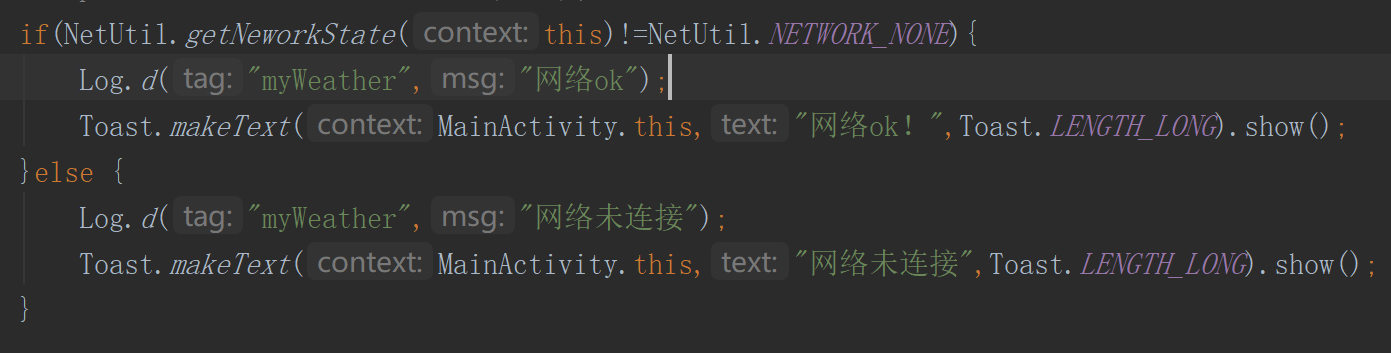
判断是否联网以及toast的使用：



一般来说MainActivity下的第一个函数都是onCreate函数：



在MainActivity中已经先定义:

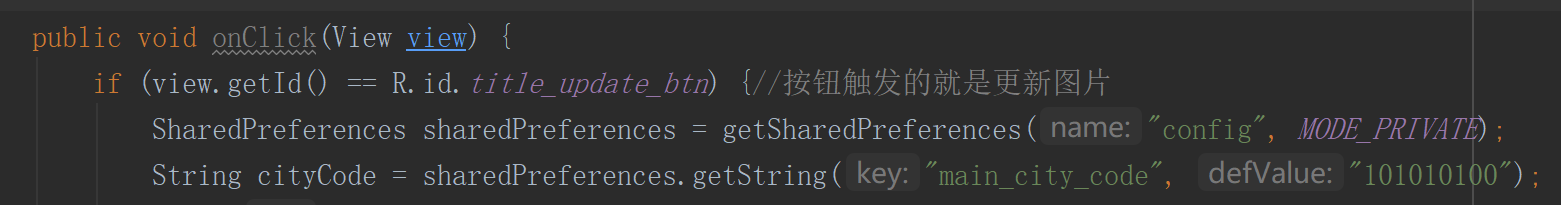
private ImageView MUpdateBtn。

所以，这三句就可以使用按钮，具体三步：

事先定义->赋值挂钩->设立监听

这三句话是onCreate函数的八股构成，一般不变

按钮点击响应函数：



SharedPreferences类，它是一个轻量级的存储类，特别适合用于保存软件配置参数。

SharedPreferences保存数据，其背后是用xml文件存放数据，文件存放在/data/data/<package name>/shared\_prefs目录下

getSharedPreferences(name,mode)方法解析：

方法的第一个参数用于指定该文件的名称，名称不用带后缀，后缀会由Android自动加上；

方法的第二个参数指定文件的操作模式，共有四种操作模式。

MODE\_PRIVATE: 私有方式存储,其他应用无法访问

利用sharedPreference类获取数据的方法：

SharedPreferences share=getSharedPreferences("Acitivity",Activity.MODE\_WORLD\_READABLE);

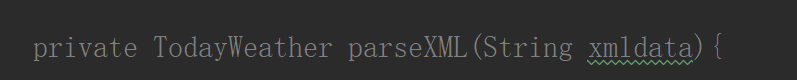
int i=share.getInt("i",0);

String str=share.getString("str","");

boolean flag=share.getBoolean("flag",false);

getString()第二个参数为缺省值，如果preference中不存在该key，将返回缺省值

定义的函数名前面加类名表示：这个函数返回的是一个这个类的实例，这个函数括号中的参数就是这个类中类型的参数，如下：



try{  
//代码区  
}catch(Exception e){  
//异常处理  
}  
代码区如果有错误，就会返回所写异常的处理。

try catch 是捕捉try部分的异常，当你没有try catch的时候，如果出现异常则程序报错，加上try catch，出现异常程序正常运行，只是把错误信息存储到Exception里，所以catch是用来提取异常信息的

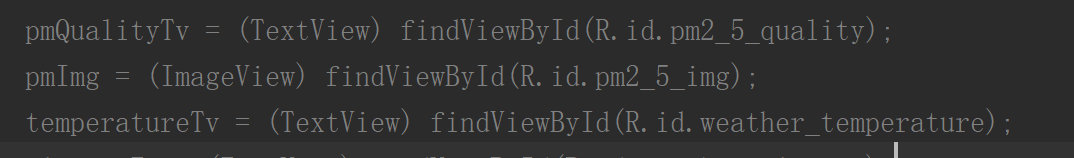
try catch

try{

……

}catch (XmlPullParserException e) {  
 e.printStackTrace();  
} catch (IOException e) {  
 e.printStackTrace();  
}

文本和图片控件也可以像数赋值操作或者指针一样，指向新的标签：

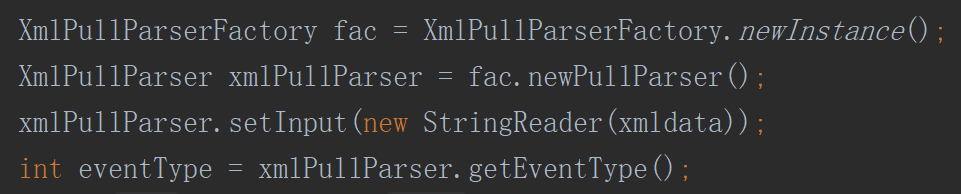


这几个是新定义的private TextView 和ImageView，可以直接用之前存在的指向标签。

并且可以使用setText的方法

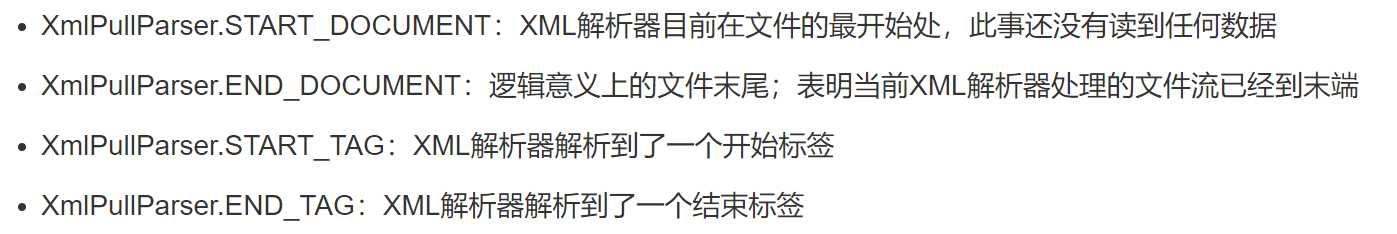


解析xml格式的数据：

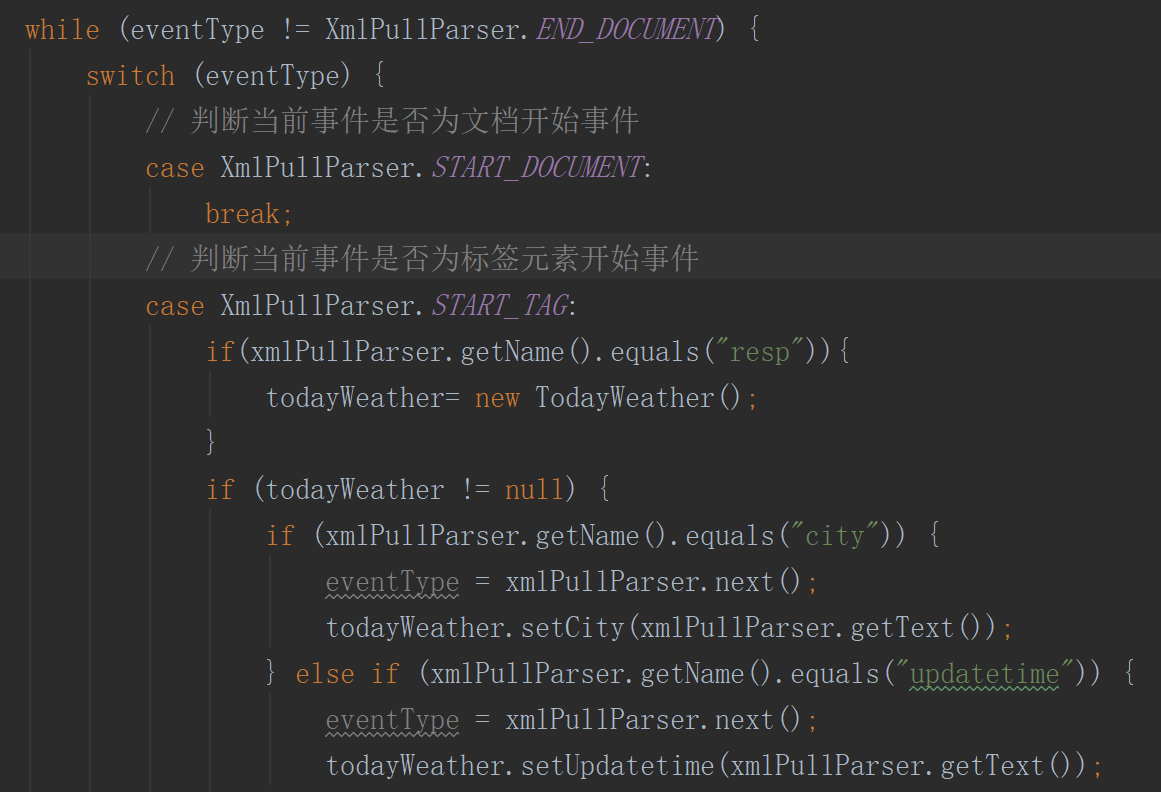


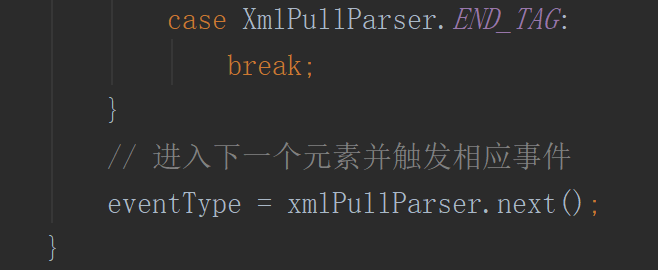
1. 首先获取到一个xmlPullParserFactory的实例
2. 然后利用这个实例获取得到XmlPullParser对象
3. 然后调用XmlPullParser的setInput（）方法将服务器返回的xml数据设置进去就可以开始解析
4. 通过getEventType（）可以得到当前的解析事件
5. 然后在一个while循环中不断进行解析，如果当前的解析事件不等于XmlPullParser.END\_DOCUMENT,说明解析工作还没有完成，调用next（）方法可以获取下一个解析事件
6. 在while循环中，我们通过getName（）方法得到当前节点的名字，满足设定的条件后，就调用nextText（）方法来获取节点内具体的内容

解析事件所得到的EventType



While循环的使用：





每次break之后，取下一个事件

多线程的使用：

使用匿名类的方式，创建新线程：

new Thread (new Runnable()){

@override

public void run(){

//具体的事务处理逻辑

}

}).start();

使用http协议访问网络：

URL url = new URL(“http://www.baidu.com”);

HttpURLConnection connection= (HttpURLConnection)url.openConnection();

Connection.setRequestMethod(“GET”);

首先需要获取到HttpURLConnection的实例，一般只需new出一个URL对象，并传入目标的网络地址，然后调用一下openConnection()方法即可

在得到HttpURLConnection的实例之后，设置HTTP请求所使用的方法，常用的方法有两个：

GET：表示从服务器那里获取数据

POST：表示提交数据给服务器

connection.setConnectTimeout(8000);  
connection.setReadTimeout(8000);

接下来自由定制，比如：设置连接超时、读取超时的毫秒数等等

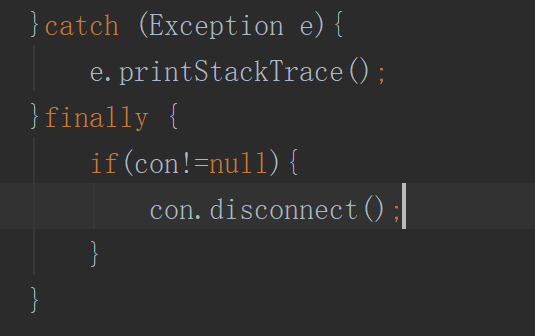
之后再调用getInputStream()方法就可以获取到服务器返回的输入流了

InputStream in=con.getInputStream();

最后调用disconnect()方法将这个HTTP协议关闭掉

connection.disconnect():

Ps：

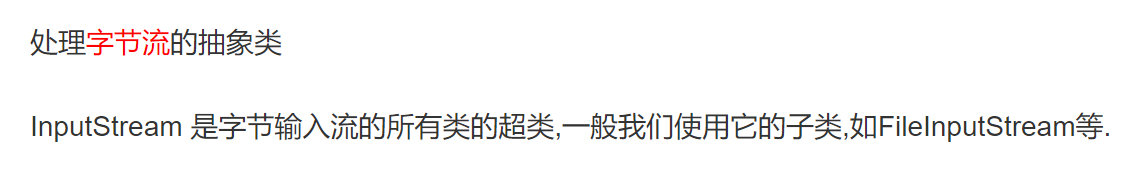


一般情况下，在try catch的最后加上finally语块来关闭连接。finally作为异常处理的一部分，它只能用在try/catch语句中，并且附带一个语句块，表示这段语句最终一定会被执行（不管有没有抛出异常），经常被用在需要释放资源的情况下。

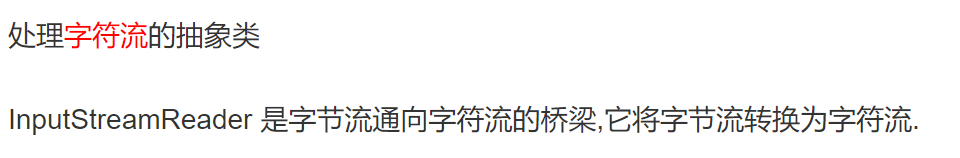
关于数据流的处理：

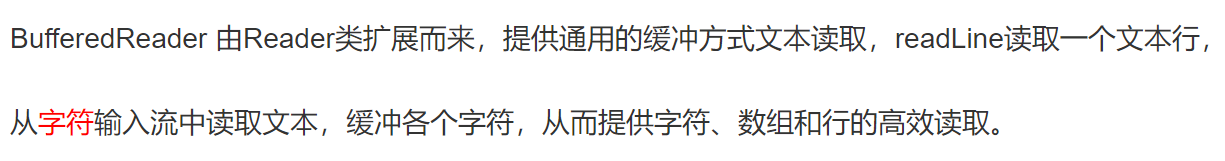


首先利用getInputString()方法获取到服务器返回的数据流，将之存储于字节流in中



然后将数据流的字节流转化为字符流



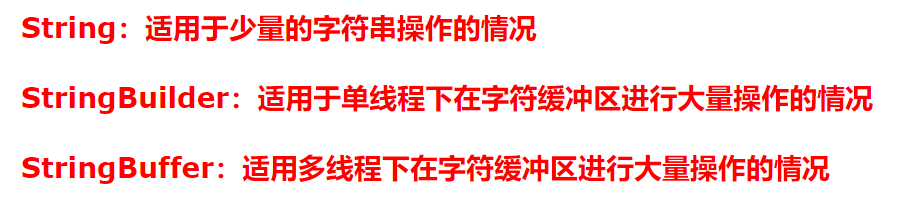


然后从字符流中读取文本，缓冲各个字符

总结如下：

InputStream能从來源处读取一個一個byte,所以它是最低级的，

InputStreamReader封裝了InputStream在里头, 它以较高级的方式,一次读取一个一个字符BufferedReader则是比InputStreamReader更高级,它封裝了StreamReader类,一次读取取一行的字符



最后利用StringBuilder在字符缓冲区下进行操作，相当于重新进行字符组合。

异步消息处理机制：

private Handler mHandler = new Handler() {  
 public void handleMessage(Message msg) {  
 switch (msg.what) {  
 case *UPDATE\_TODAY\_WEATHER*:  
 updateTodayWeather((TodayWeather) msg.obj);  
 break;  
 default:  
 break;  
 }  
 }  
}

这一步是在主线程中完成：

1. 创建Handler对象
2. 写hanleMessage函数，其中包括收到消息后的事务处理逻辑

Handler，用于发送和处理消息，发送消息用Handler的sendMessage()方法，消息最终传递到Handler的handleMessage()方法中。

这一步是在子线程完成：

1. 创建msg对象
2. 设定what字段
3. 设定obj字段
4. 发送消息

Message的what字段一般是一个整型常量，需要赋值

obj字段可以用来携带一个object对象

Message msg =new Message();  
msg.what = *UPDATE\_TODAY\_WEATHER*;  
msg.obj=todayWeather;  
mHandler.sendMessage(msg);

项目框架总结：

1. 先设计出app的UI布局
2. 用onCreate函数加载布局，设置更新按钮。随后写出onClick函数来设定点击按钮的响应操作
3. 新建一堆TextView和ImageView控件来指向当初设计布局时对应的TextView和ImageView控件，这一操作的目的是后期可以很方便地使用新建控件的setText()方法来改变控件中的内容
4. 新建一个类，这个类中的一堆属性就对应着城市名、湿度、温度等等，而这个类的set()方法和get()方法分别可以将类的各个属性赋值和取值。
5. 定义解析函数，解析函数的目的就是将网络获取的数据流解析出来，然后用set()方法将上述的类的各个属性分别赋值
6. 定义更新函数，目的就是先利用get()方法将已经赋值完毕的类的各个属性的值取出来，然后利用新建控件的setText()方法将这些值覆盖原来的text内容，达到更新的效果

函数调用顺序：

一旦点击更新的图片按钮，触发onClick()函数

onClick函数中调用了天气查询函数query()

query()函数中先是利用HTTP协议获取到数据流，然后调用了解析函数parse()进行了解析并对TodayWeather类的各个属性进行了赋值，随后发送了消息

主线程收到了消息，就在更新函数update()中利用发送消息中携带的TodayWeather对象进行了控件内容的覆盖，实现更新！