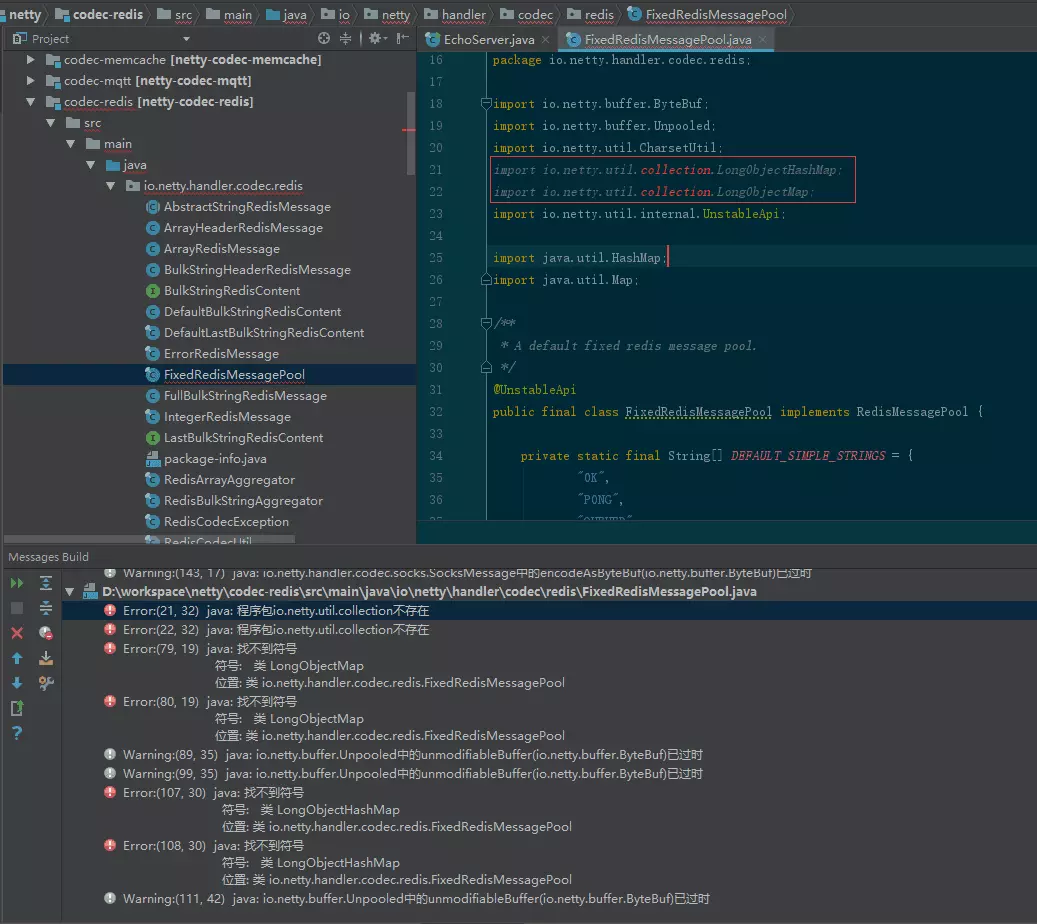
https://www.jianshu.com/p/9160684f134b

**Netty 分享之动态生成重复性的代码**

0.062018.02.09 15:58:02字数 858阅读 1283

**逅弈** 转载请注明原创出处，谢谢！

我在调试Netty提供的example时，IDEA build完后报了一些类不存在的错误，比如其中有一个报缺少LongObjectHashMap的错：



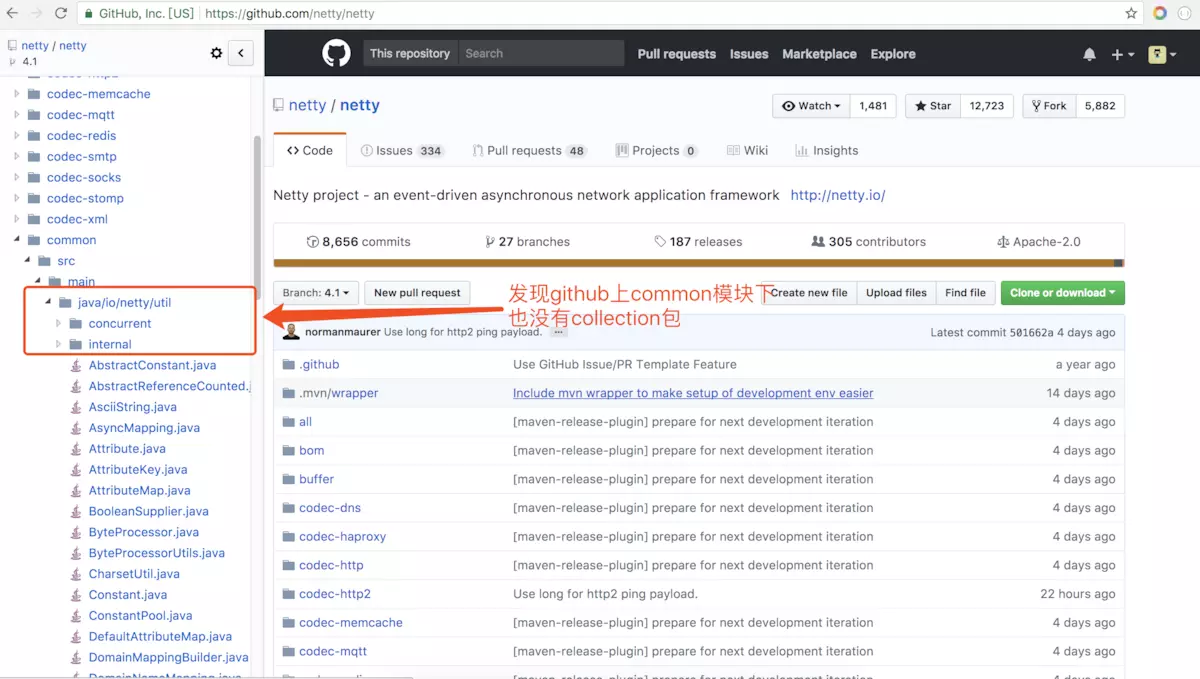
img1.png

到common模块下发现io.netty.util包下确实没有collection包:



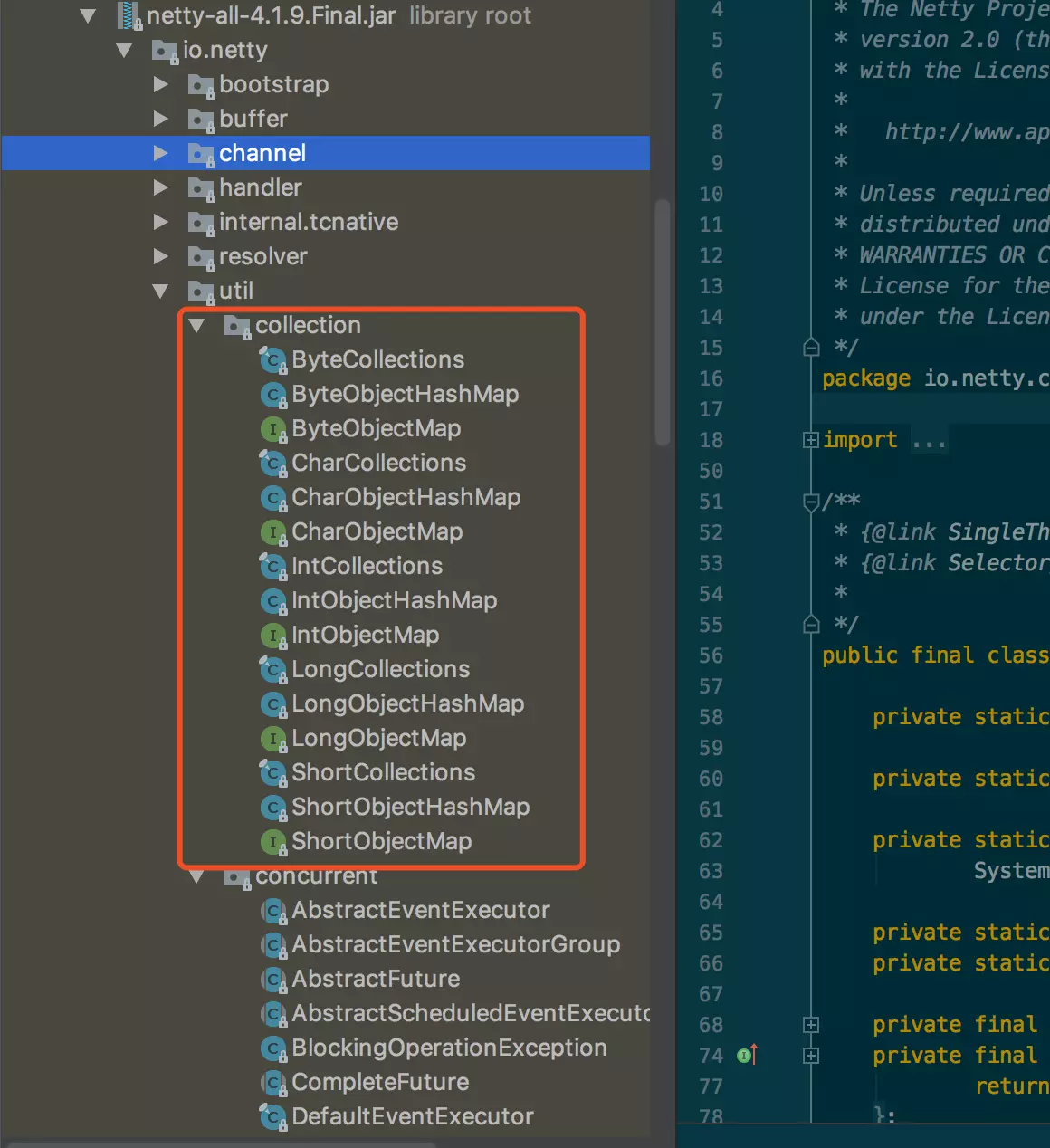
img2.png

缺少了代码，为了让程序能成功运行起来，肯定要把这缺少的代码找回来啊。首先我用git pull origin 4.1命令再一次pull了一把，但是确实没有新的代码了。然后我到github上查看了netty的源代码，发现确实也没有这个collection包：



img3.png

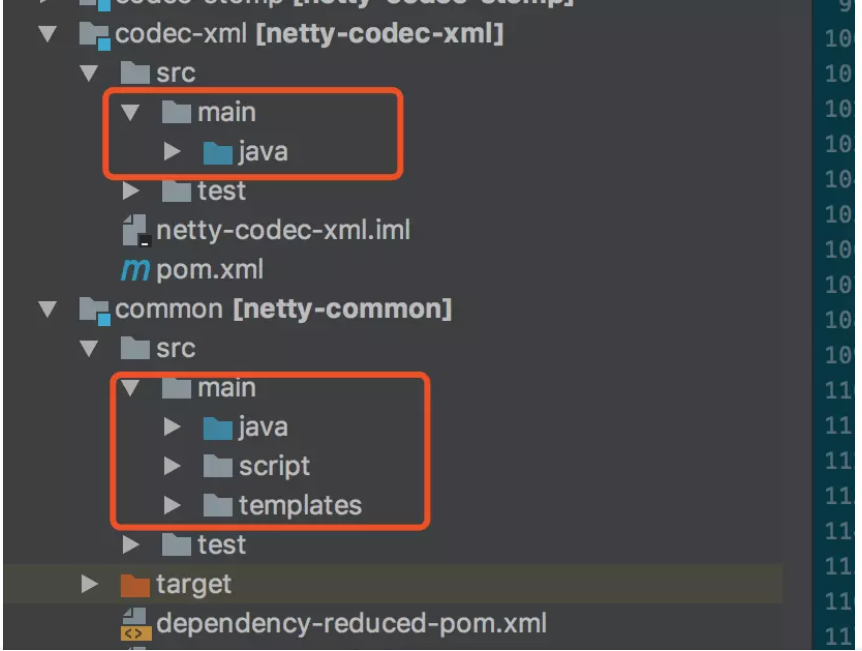
fork下来的代码肯定是没有问题的，我又到我们用到netty的项目中去找netty的jar包，在jar包里面发现是存在collection包的。其实想想jar包里面肯定是存在的，否则release的版本都没办法运行了。



img4.png

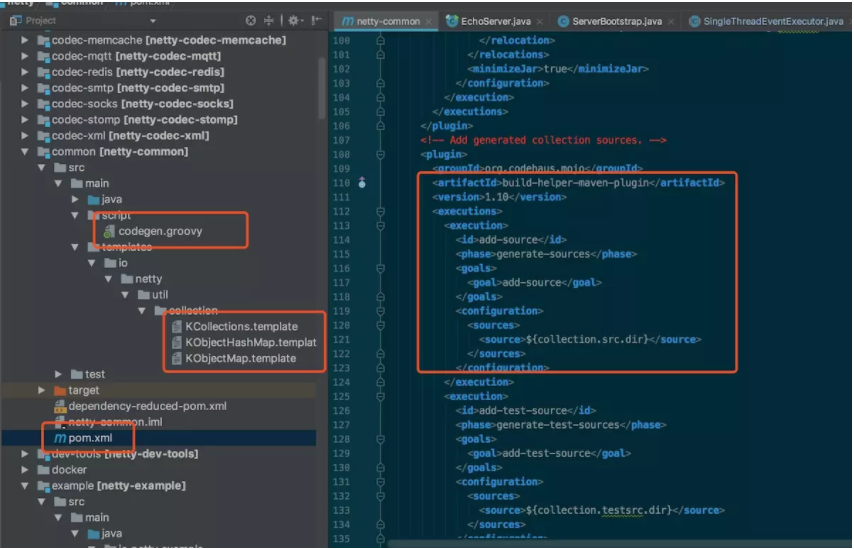
为什么会缺少collection包呢，为什么不是缺少其他的包呢？难道collection的包有什么特别之处？  
在netty-all-4.1.9.Final.jar包中可以看到，collection包下面的类很有特点，长得都差不多嘛。ByteCollections、CharCollections、ShortCollectons、IntCollections、LongCollections，都是XXCollections。其他的还有XXObjectMap，XXObjectHashMap。

带着这些疑问和线索再次到netty的源码中去寻找答案。既然这些代码是在common模块下，那就到common中去寻找吧。我发现common模块和其他的模块有些许不同的地方，多了一个script和templates目录。我当时就敏锐的觉得，肯定跟这两个目录有关，应该是通过templates目录下的模板，使用script目录下的脚本动态生成了collection包下的代码了！



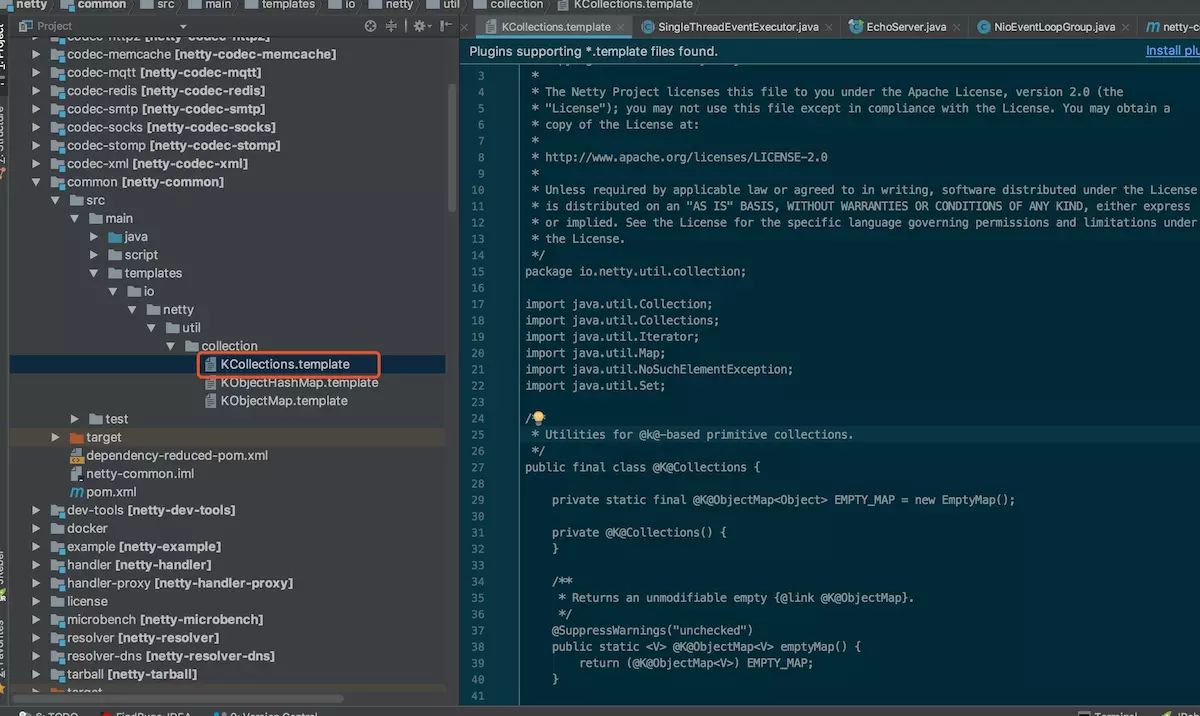
img5.png

带着疑问一层一层的剥开script和templates的目录，我发现了冰山下巨大的宝藏，collection包就是由脚本动态生成的，确定无疑了：



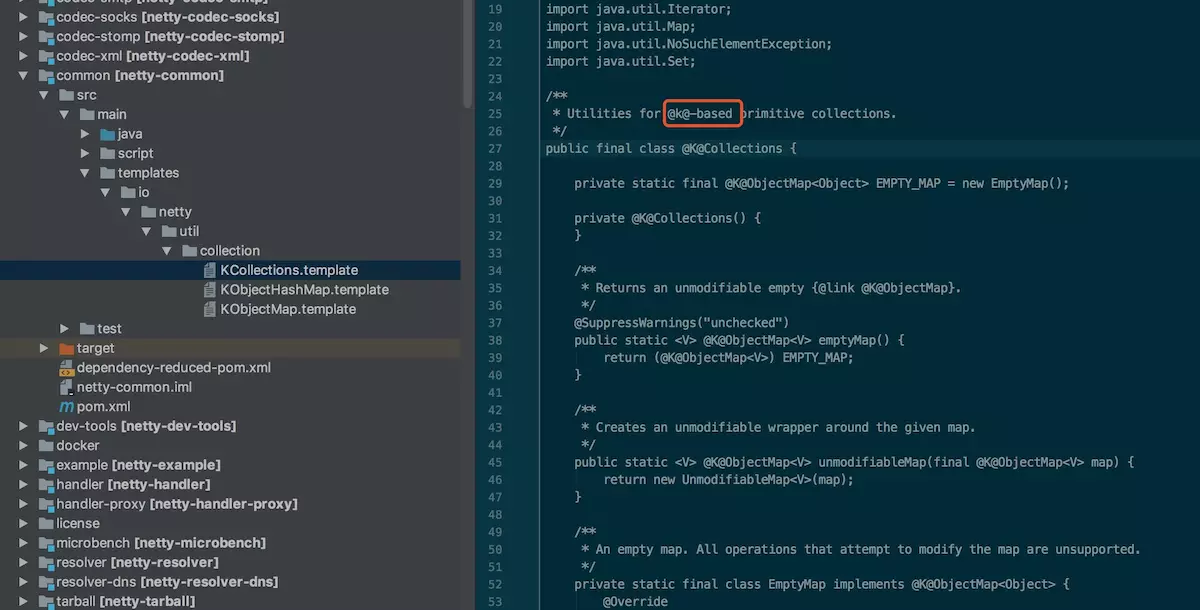
img6.png

从上图可以看出，通过maven的build-helper-maven-plugin插件，通过codegen.groovy脚本，读取templates目录下的脚本，生成了所需要的collection包下的具体的类。



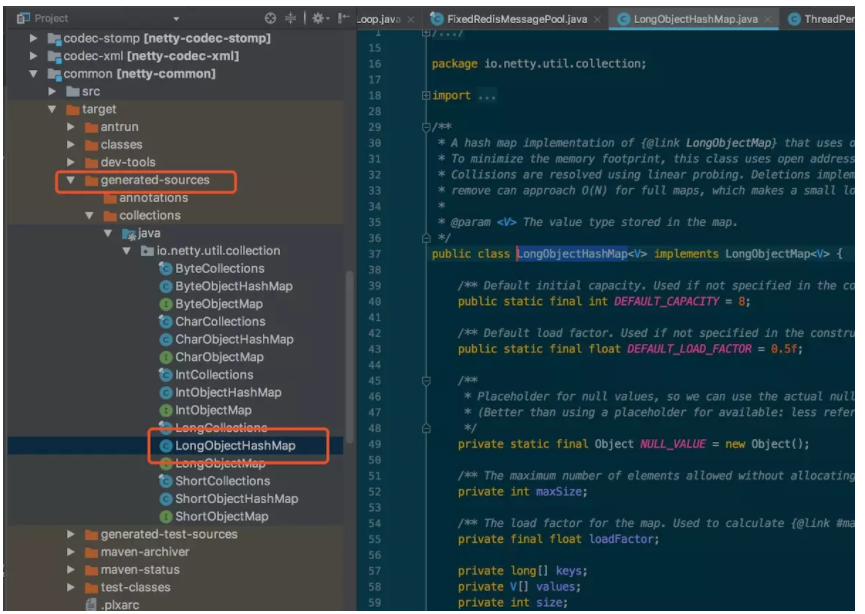
img7.png

看一下KCollections.template模板的内容，其中有一个占位符@K@，表示这是一个基于K的Collection。



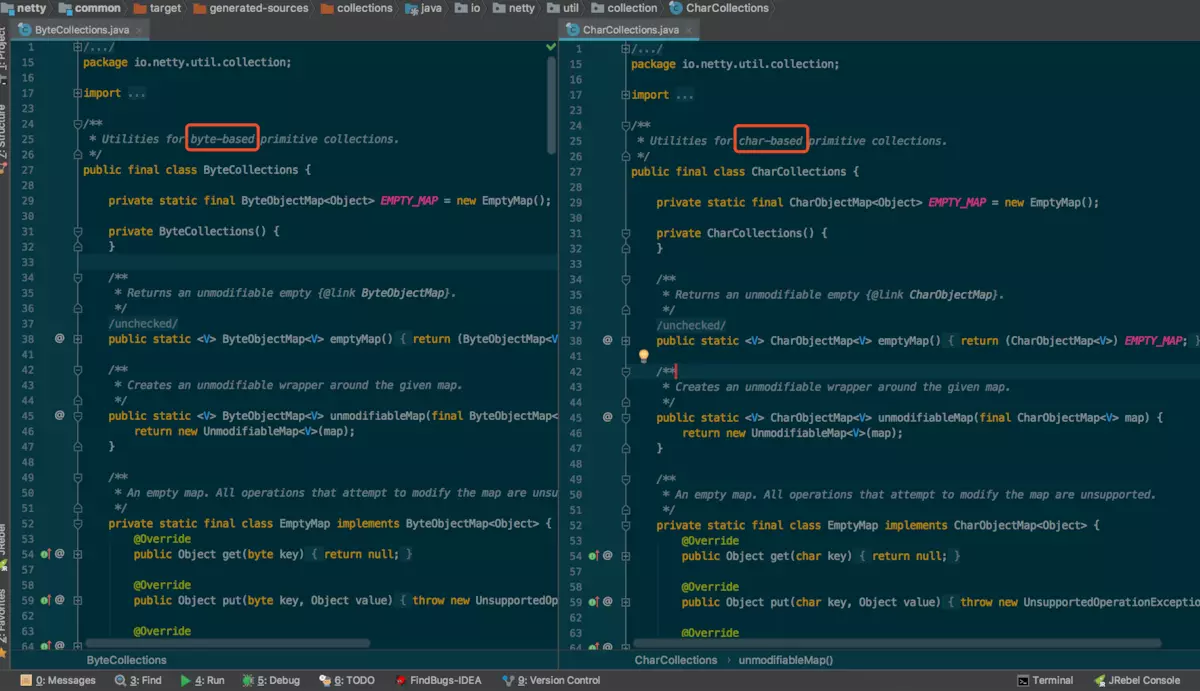
img8.png

最终生成的代码会在generated-sources目录下：



img9.png

拿两个类比较一下，发现他们除了K不同外，其他都一样，这也就证实了一个问题，为什么要把collection包下的代码放到模板里面，动态生成呢？



img10.png

其实，我们在实际的开发的过程中也遇到过这样的问题：两个类(甚至是更多的类)的功能都一样，不一样的只是他们要操作的其中的一个变量或者某一个对象的类型不一样。那为了固化代码，避免重新性的编写大量的代码，可以通过模板把这种会重复的类的代码固定下来，对于可能会发生变化的内容通过占位符的形式(如上面例子中的@K@)表示，最终当需要用到某种类型的类时，指定K来动态生成就好了。

进入common下，执行 clean package -Dcheckstyle.skip=true    会在${project.build.directory}/generated-sources/collections/java目录下自动生成代码，再编译就不报错了



更多原创好文，请关注「逅弈逐码」