**知识点：mybatis多数据源**

**\*DataSource本身是接口，有getConnection()方法，就是返回一个数据库连接；**

**\*** **ThreadLocal**

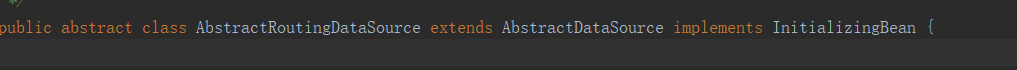
不是线程，而是线程的一个变量，为当前线程设置共享变量值，为其他线程使用提供独立的副本，不通线程只能通过set,get，remove自己的变量副本，而不影响其他线程的副本，可以解决多线程并发；详见thread文档

**\* AbstractRoutingDataSource抽象类**

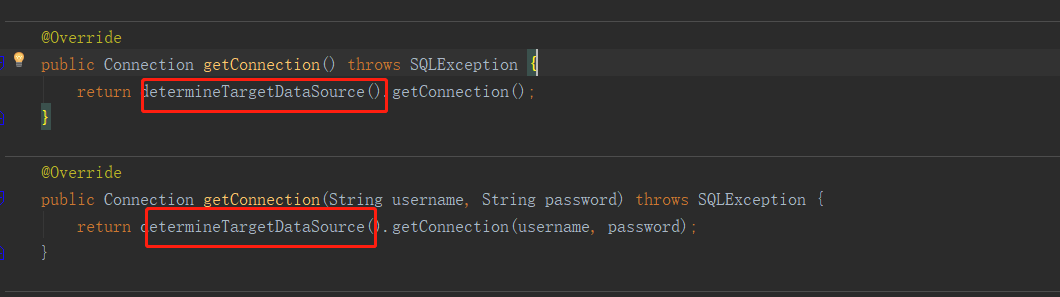
动态切换数据源，多数据源用到

**\*切面编程**

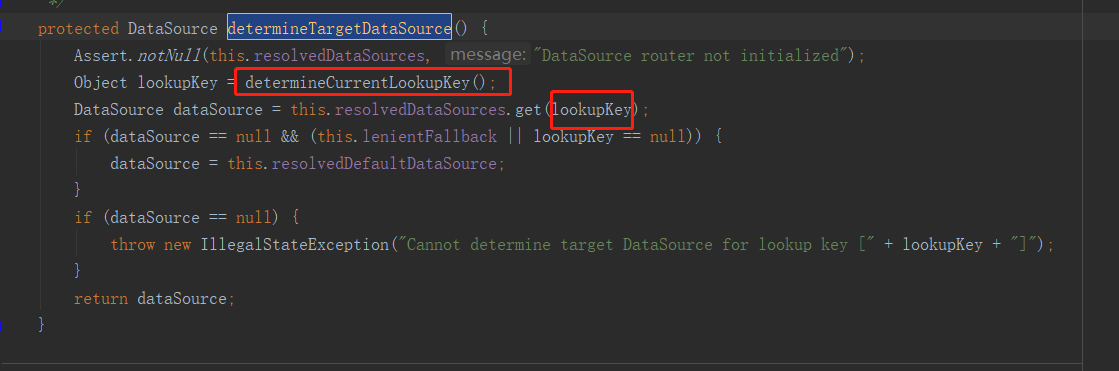
1. **AbstractRoutingDataSource抽象类：动态切换数据源，多数据源用到**



实现了AbstractDataSource,而抽象类AbstractDataSource实现了DataSource接口；所以我们看AbstractDataSource的于是我们自然地回去看它的getConnection方法：

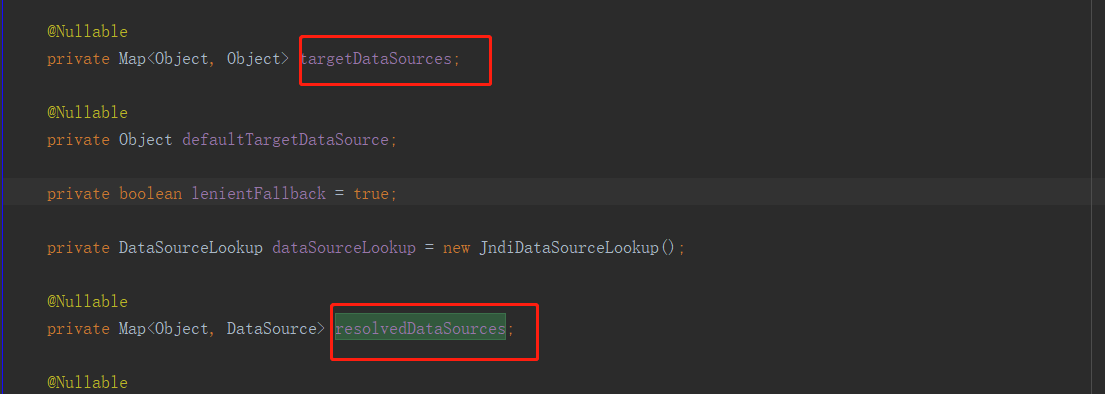


原来关键就在determineTargetDataSource()里：我们再看determineTargetDataSource()



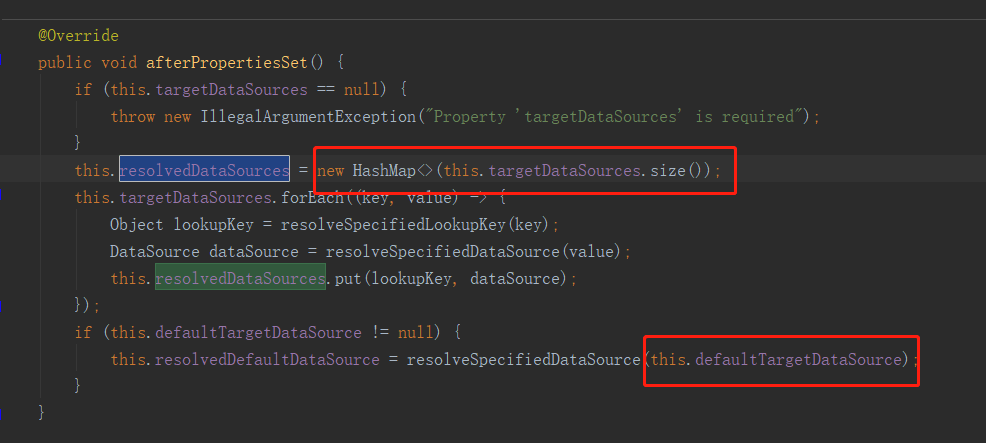
决定用哪个数据源是determineCurrentLookupKey（）决定的，它返回的是一个object，作为key值去resolvedDataSources该map集合变量取数据源，所以我们看resolvedDataSources的定义；（或者resolvedDataSources为空时返回resolvedDefaultDataSource：此属性也应该在该抽象类子类作为配置类时，初始化的时候set进去属性，默认的数据源；

）

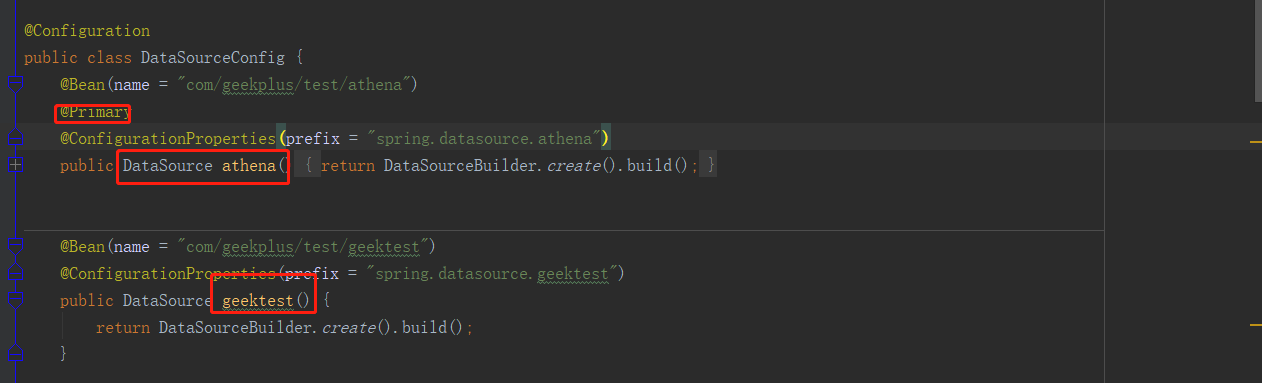


targetDataSources，resolvedDataSources都是储存datasouce的map集合，

我们先看resolvedDataSources是怎么把数据源放进去的，然后看determineCurrentLookupKey（）怎么获取数据源的key的，



因为此类实现了InitializingBean接口，它定义了afterPropertiesSet方法，实现它的子类有个特点，会在子类的属性初始化完成后，也就是这个类被@bean注解的方法执行完后（经常在配置类中用），才会调用；看源码，resolvedDataSources数据源集合来源于两个属性一个是targetDataSources ，一个是defaultTargetDataSource；所以在**AbstractRoutingDataSource抽象** 类的子类（下面的DynamicDataSource）在作为配置类时，注册为bean之前就要给它俩赋值，也就是你需要的数据源都给赋值进去，所以新建配置类DataSourceConfig.java如下图：

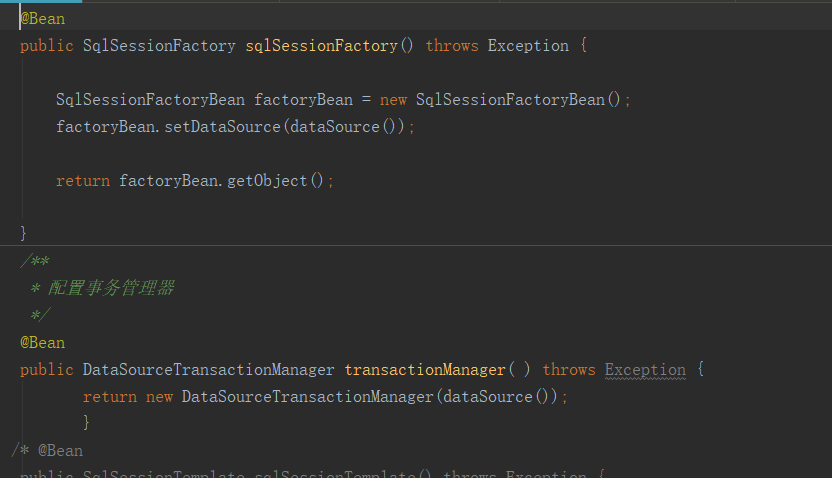


包括主数据源，和测试数据源，还有下图的动态数据源对象，其实也是datasource，设置默认数据源，目标数据源，目标数据源里有可动态切换的所有数据源，

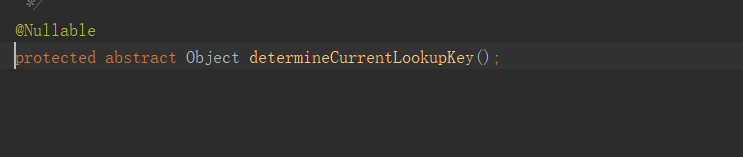


还有sqlsesstion把两个数据源都加进去，实现具体sql事务的操作；

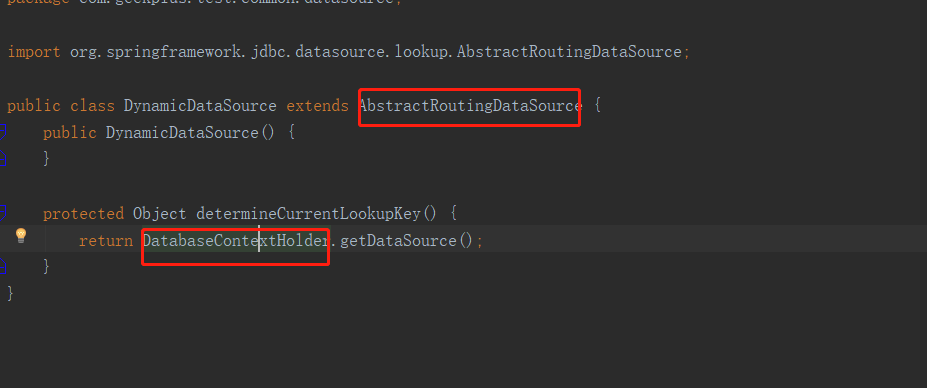
以及事务管理也要加入两个数据源



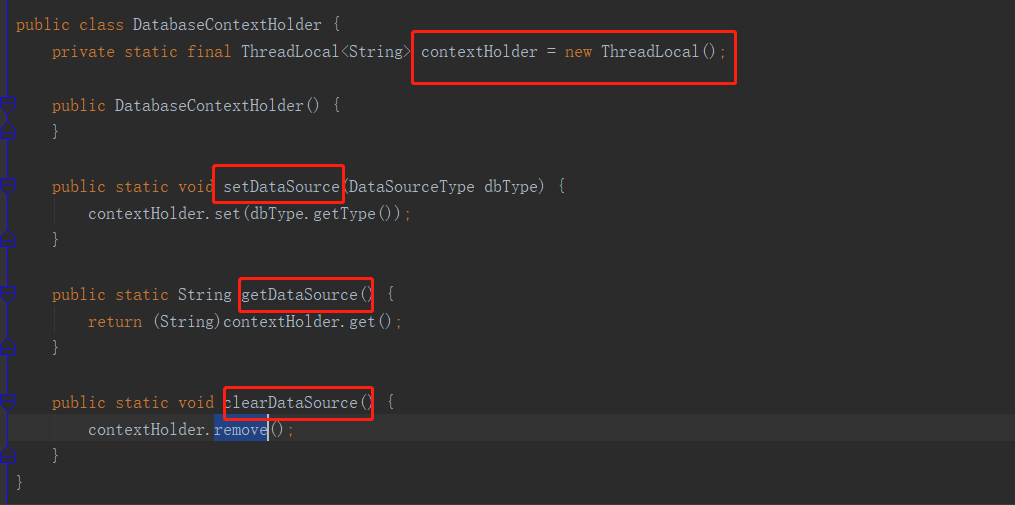
下面还是接着看AbstractRoutingDataSource类的决定连接哪个数据源的 determineCurrentLookupKey（）方法，



是个抽象方法。也就是说必须新建个子类继承AbstractRoutingDataSource并重写它，来获取具体数据源对应的名字；那我们新建DynamicDataSource类(动态数据源类)实现determineCurrentLookupKey方法，（并且用它作为生成配置类bean赋值）

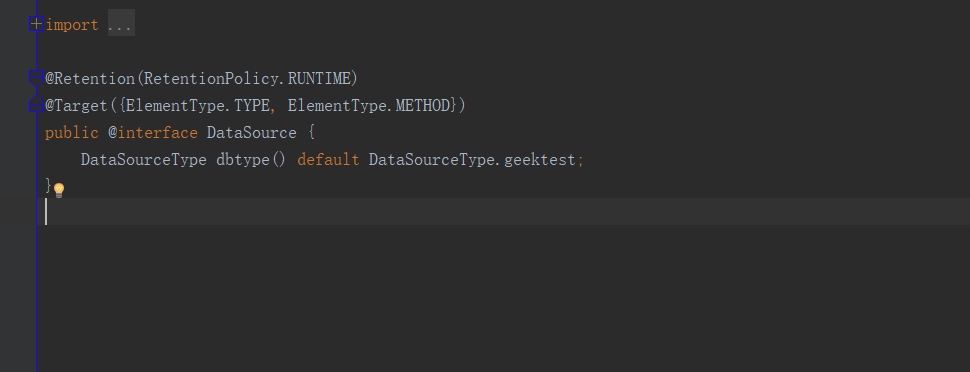


返回一个要使用的数据源名称，DatabaseContextHolder（数据库上下文的 的持有者）类就是用到ThreadLocal来实现动态把数据源名称保存起来多数据源是以枚举类型保存， 我们编写DatabaseContextHolder ，

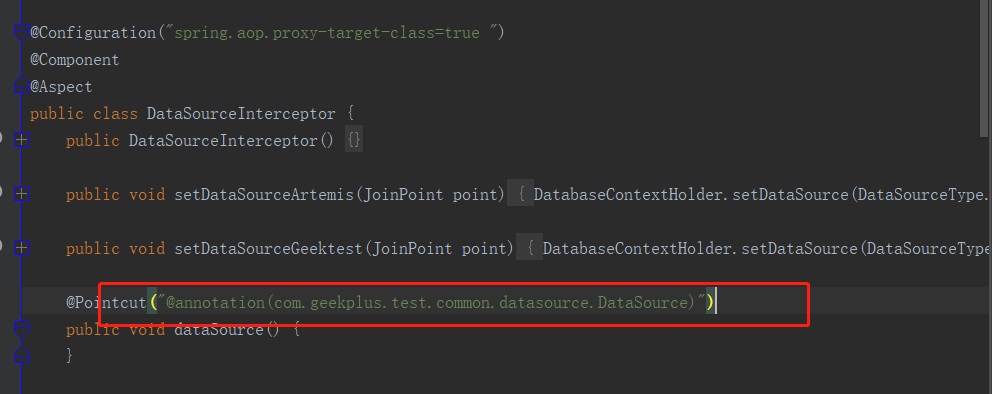


然后新建一个和一个自定义注解DataSource和切面DataSourceInterceptor.java切点就在注解（DataSource）上，

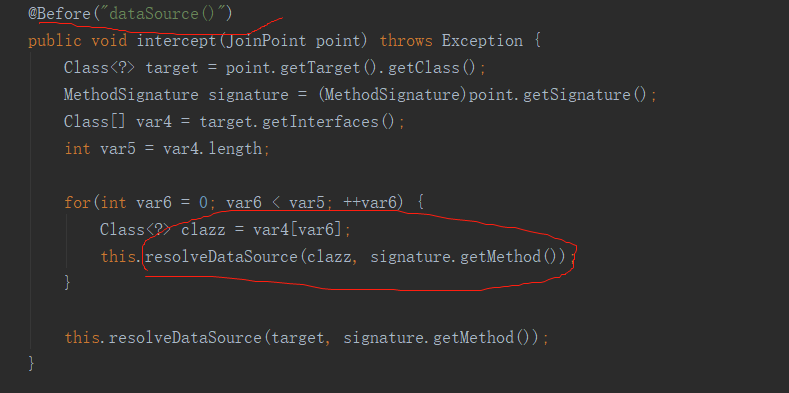
先看自定义注解DataSource，注解在方法上，运行时起作用，注解的值就是多数据源枚举值，（写哪个就是代表要用哪个）



再看切面DataSourceInterceptor，切点在DataSource自定定义注解上

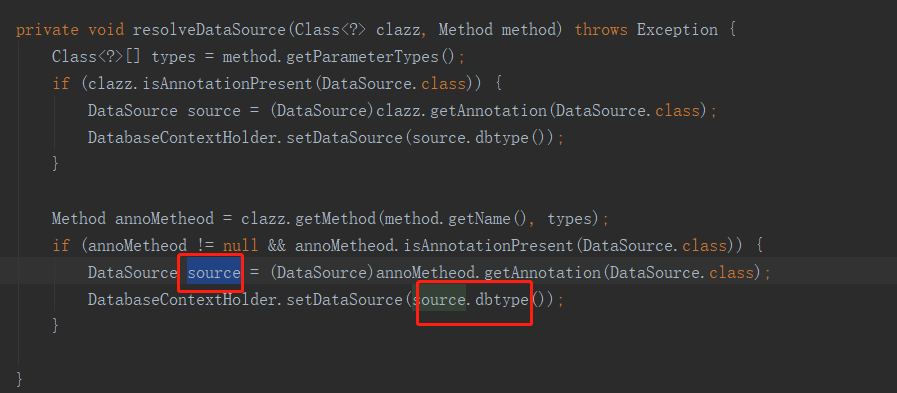


在切点执行之前，执行resolveDataSource方法提前设置当前线程的全局变量值DatabaseContextHolder. setDataSource(DataSourceType dbType)，设置要用哪个数据源；



我们看resolveDataSource就是设置的注解的值， 比如@dataSource（“athena”）

返回的注解值就是athena；



在执行完自定义注解的方法后执行，清除当前线程中的变量数据源名称

