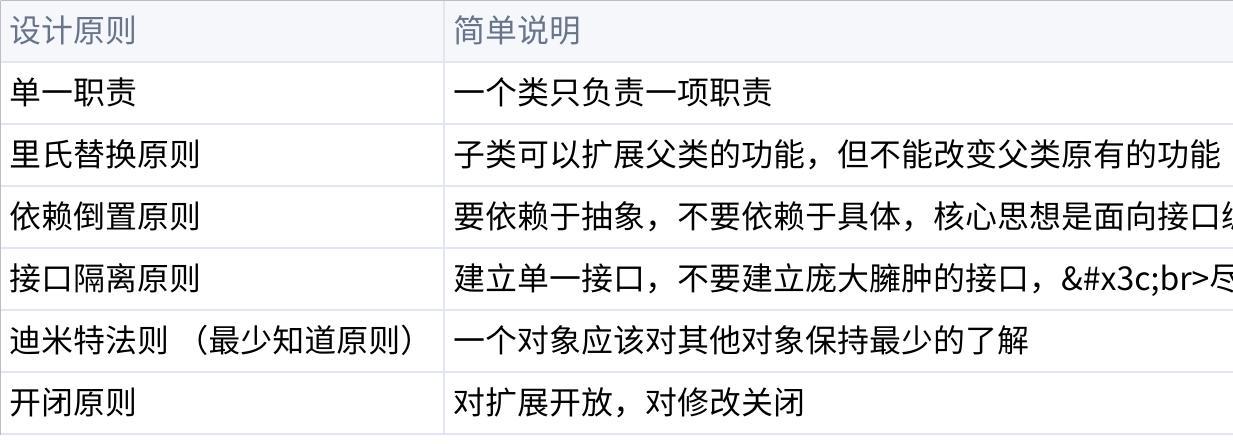
**设计模式**

**设计模式篇**

**1.谈谈你对设计模式的理解**

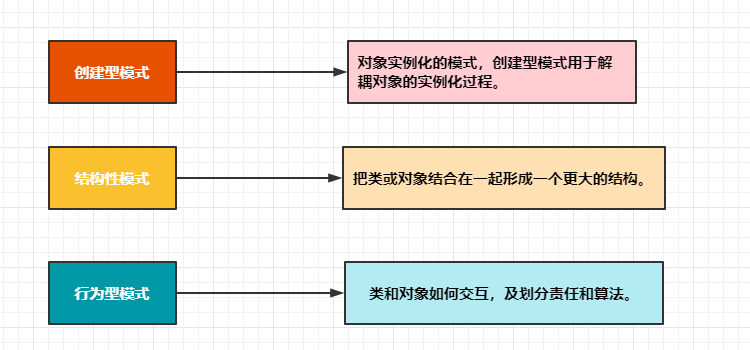
1.首先谈设计模式的作用：经验的传承，提高了软件复用的水平，最终达到提高软件开发效率

**设计模式原则**

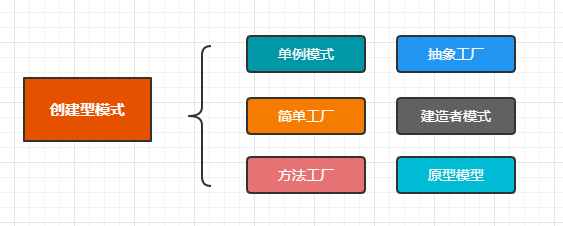


**点击图片可查看完整电子表格**

**设计模式的分类**



3.创建型模式：都是用来帮助我们创建对象的！



4.结构性模式：关注对象和类的组织



5.行为型模式：关注系统中对象之间的相互交换，研究系统在运行时对象之间的相互通信和协作，进一步明确对象的职责，共有11中模式



**2.各模式的作用**

**创建型模式**

**各个模式的作用**



**结构性模式**

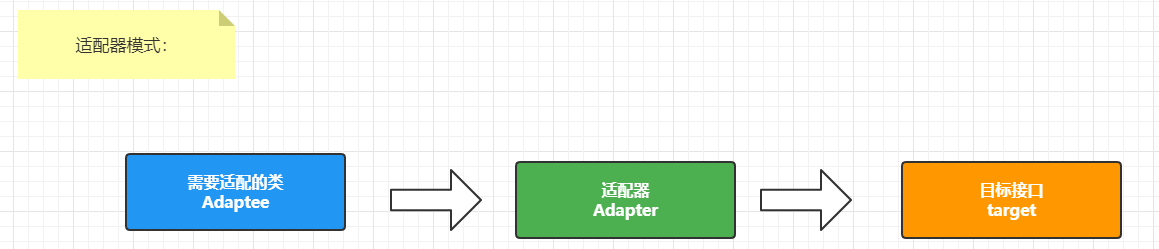


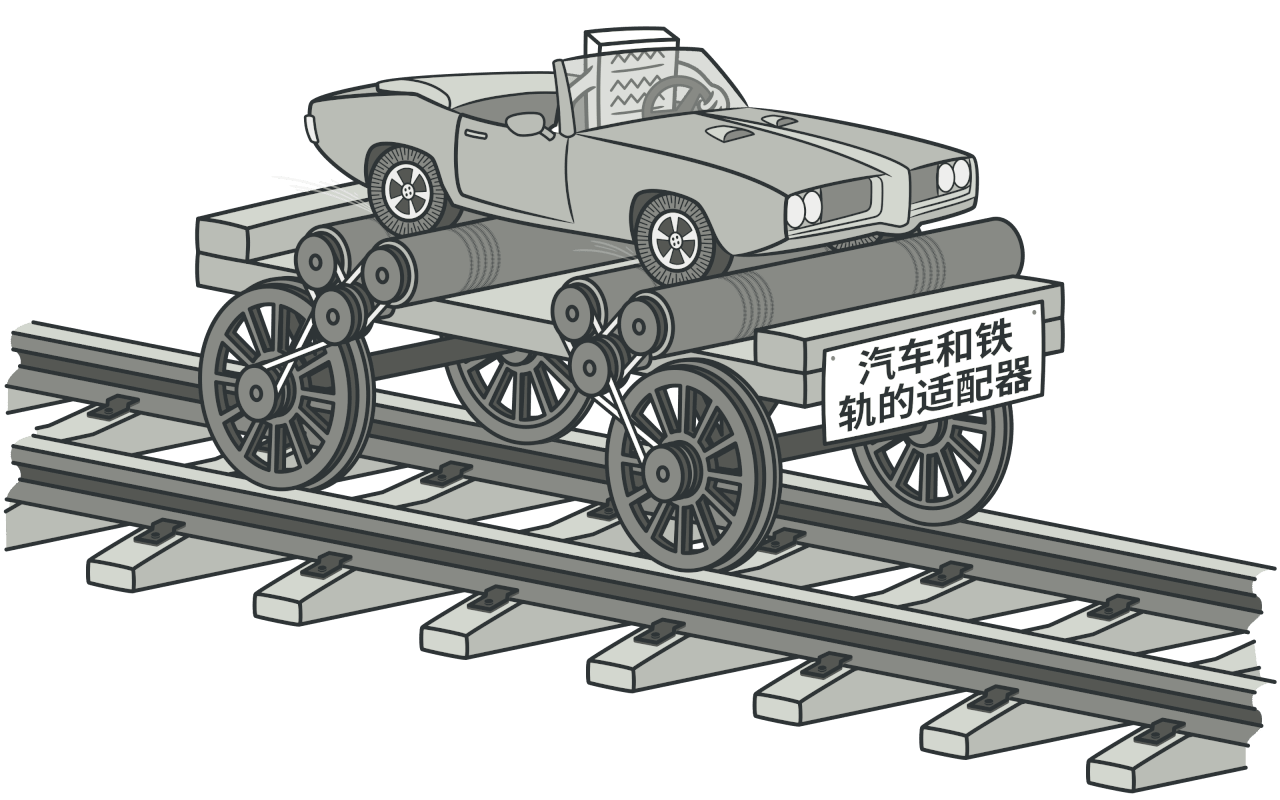
**关注对象和类的组织**

**结构型模式-适配器模式**

**4.1 适配器的作用**

  适配器模式的作用是把两个不兼容的对象通过适配器能够连接起来工作。

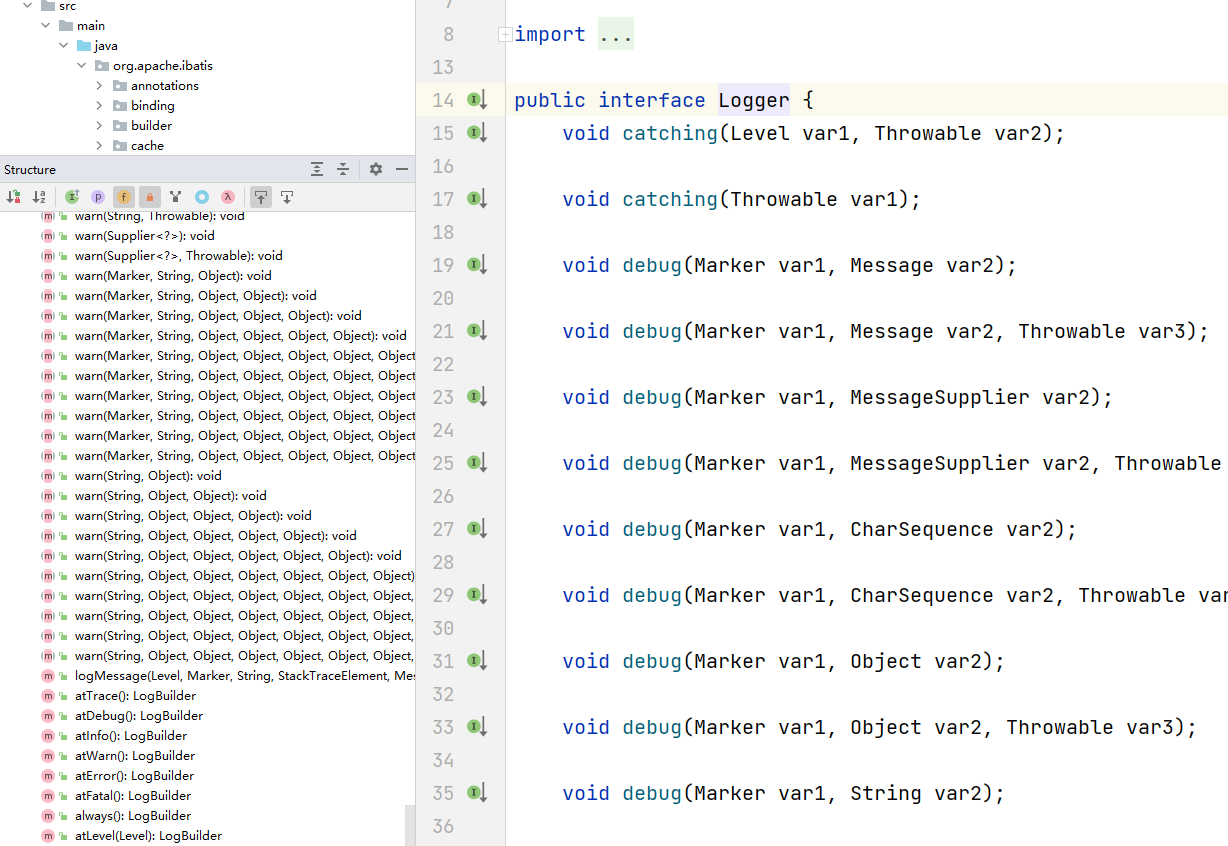




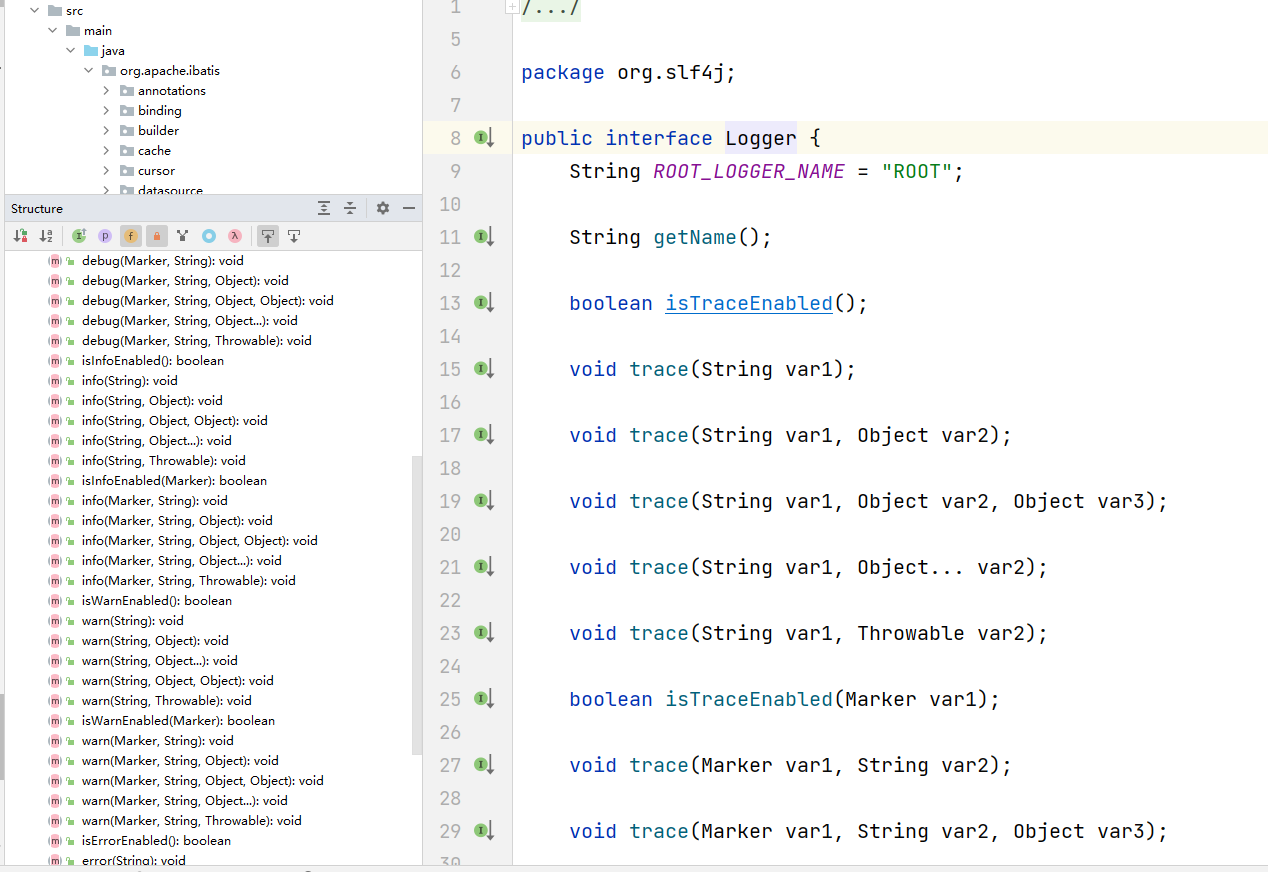
**4.2 具体案例分析**

  以 MyBatis 中的日志模块为例来介绍。常见的日志框架有 log4j，log4j2，slf4j，logbak 等，但是每种日志框架中的日志级别都有差异。

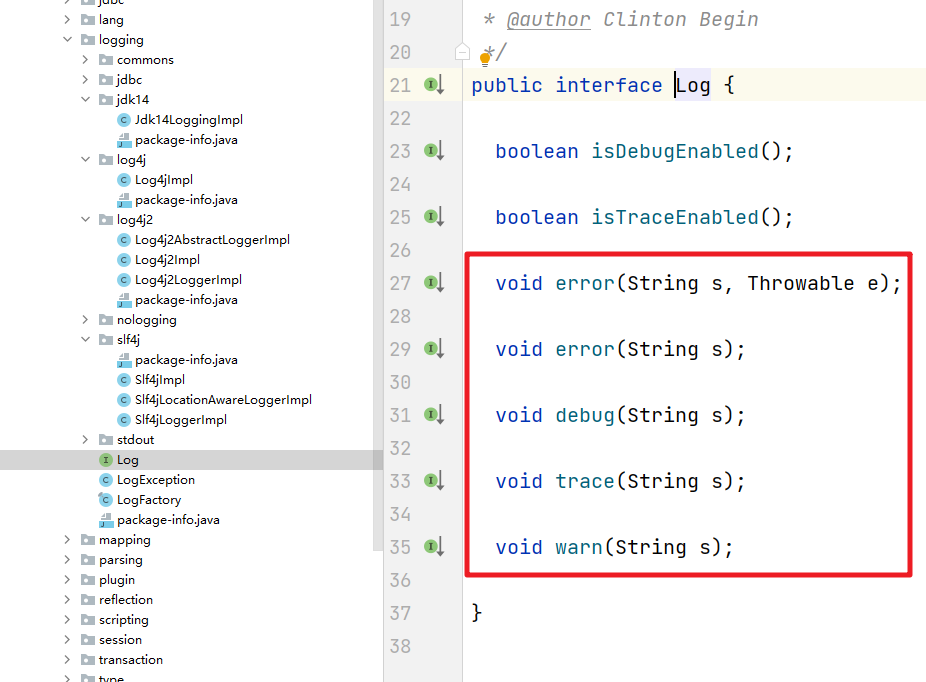
log4j2的接口：



slf4j的接口



也就是可以看到不同的日志框架里面所定义的日志级别和对应的方法都有区别，那么我们的框架怎么来统一使用这些日志框架呢？在MyBatis中通过定义了一个日志接口，定义了日志具有的级别和方法。



那这时候我们就发现具体的日志框架和这个接口其实是没有办法直接来使用的。

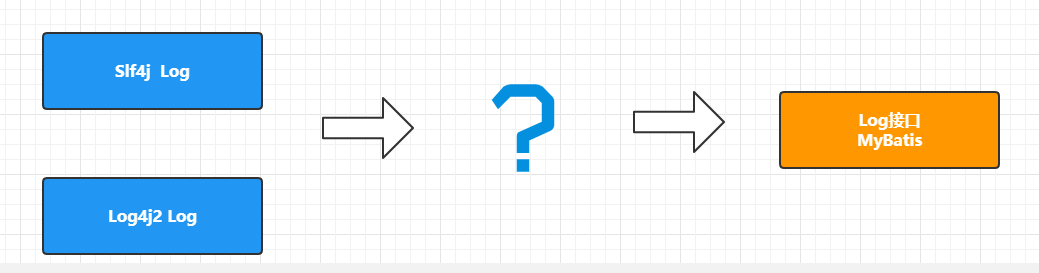
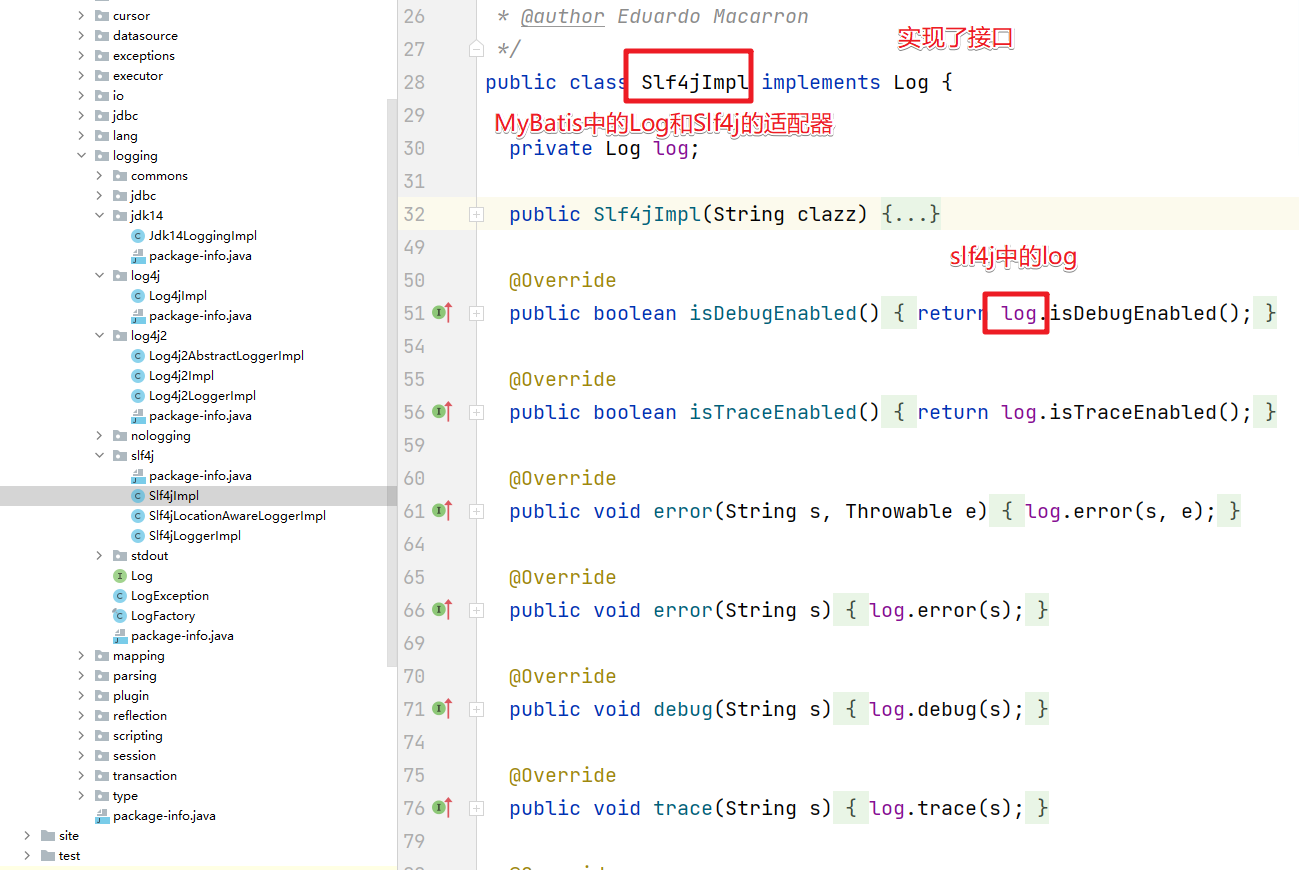


image.png

这时我们就需要通过对应的适配器来处理这种情况，以Slf4J为例。



**结构型模式-装饰者模式**

**5.1 装饰者模式的作用**

  装饰者模式又称为包装模式（Wrapper），作用是用来动态的为一个对象增加新的功能。装饰模式是一种用于代替继承的技术， 无须通过继承增加子类就能扩展对象的新功能 。使用对象的关联关系代替继承关系，更加灵活，同时避免类型体系的快速膨胀。



image.png

**5.2 装饰者模式的应用**

  装饰者模式的应用场景还是非常多的，比如

* IO流中的FileInputStream，FileOutputStream等
* Spring中的各种Wrapper
* MyBatis中的缓存设计

我们以MyBatis中的缓存实例为例来看看其具体的实现。

首先是Cache接口

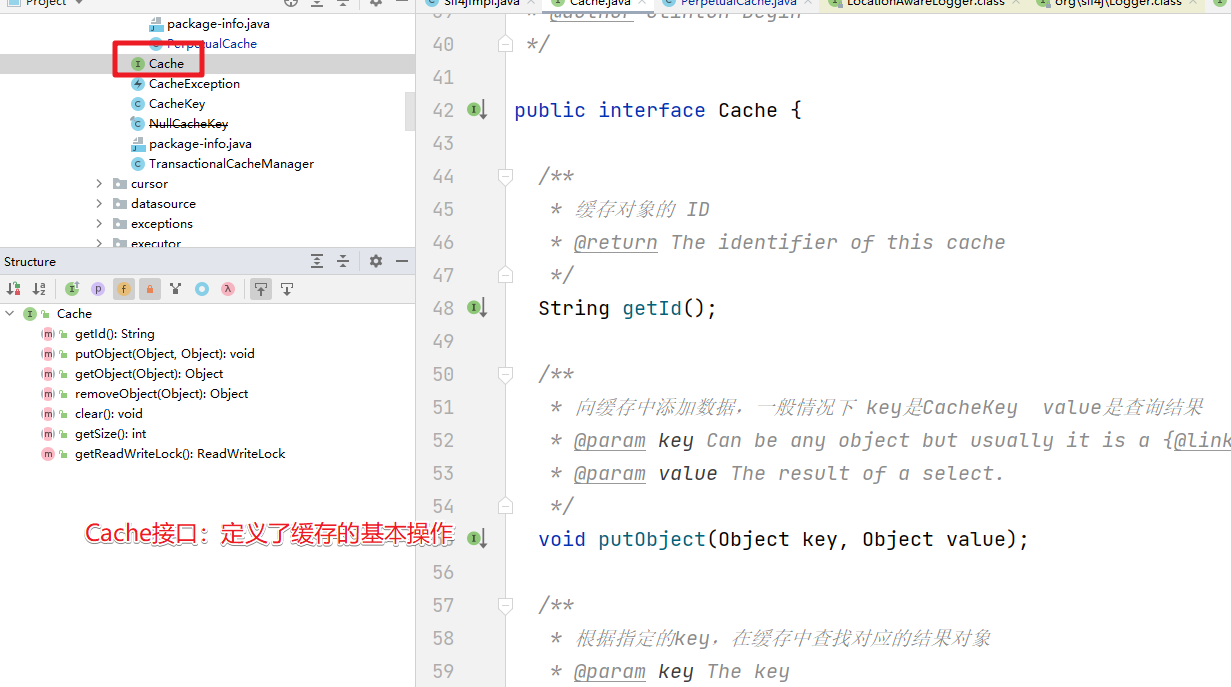


image.png

然后是PerpetualCache实现：仅仅实现了数据基于内存的读写操作。功能单一。

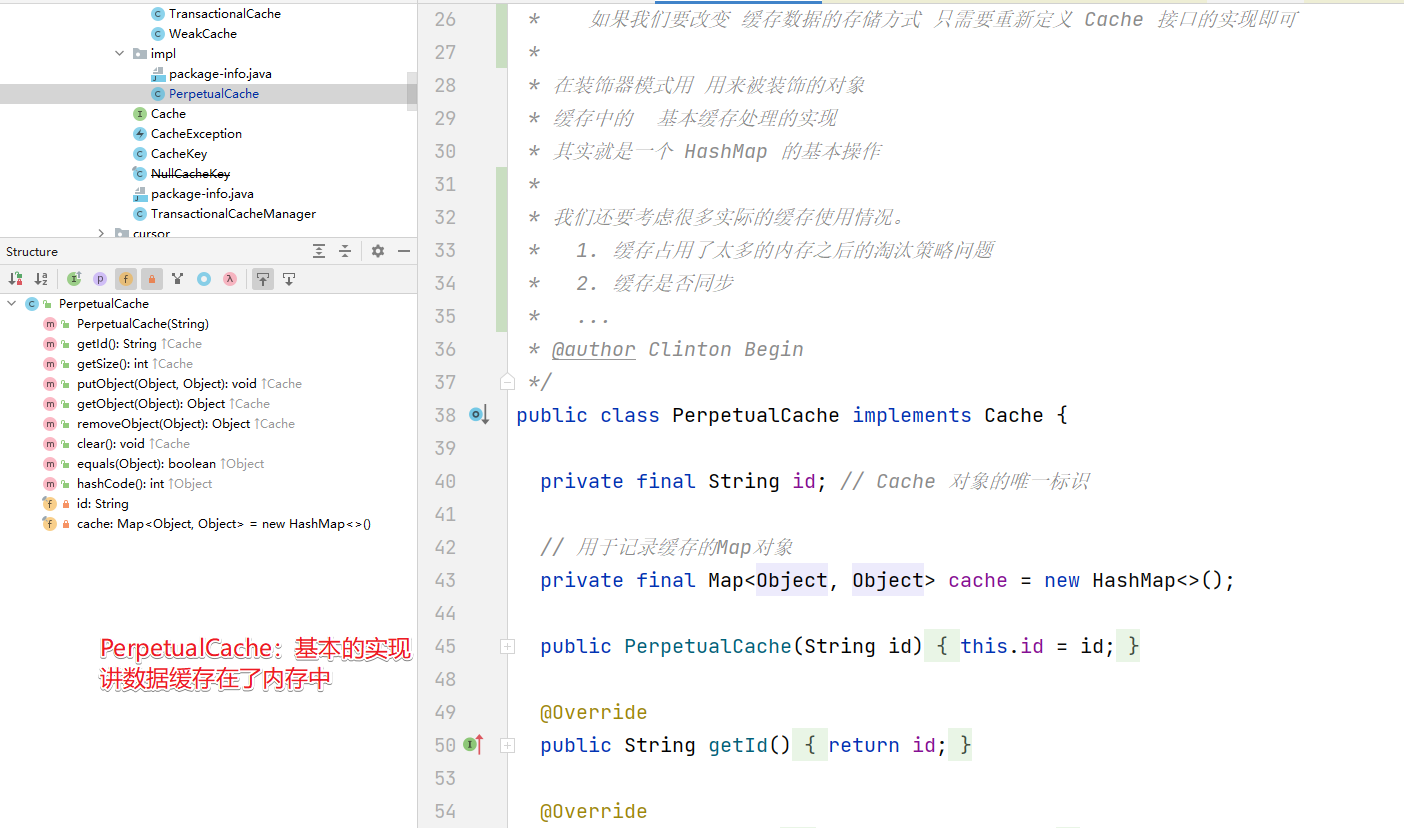


image.png

装饰类：然后在MyBatis中给我们提供了很多的装饰类。

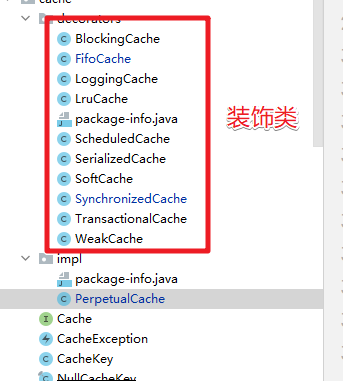
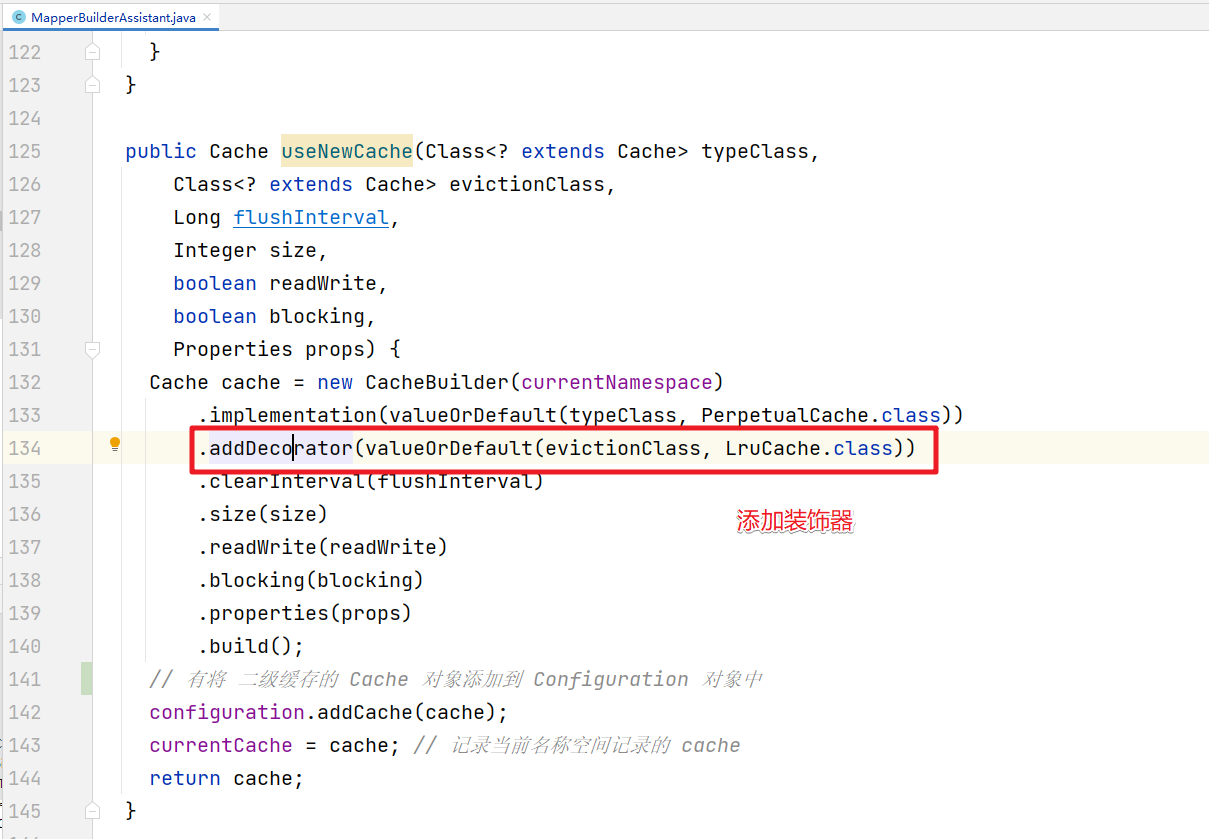


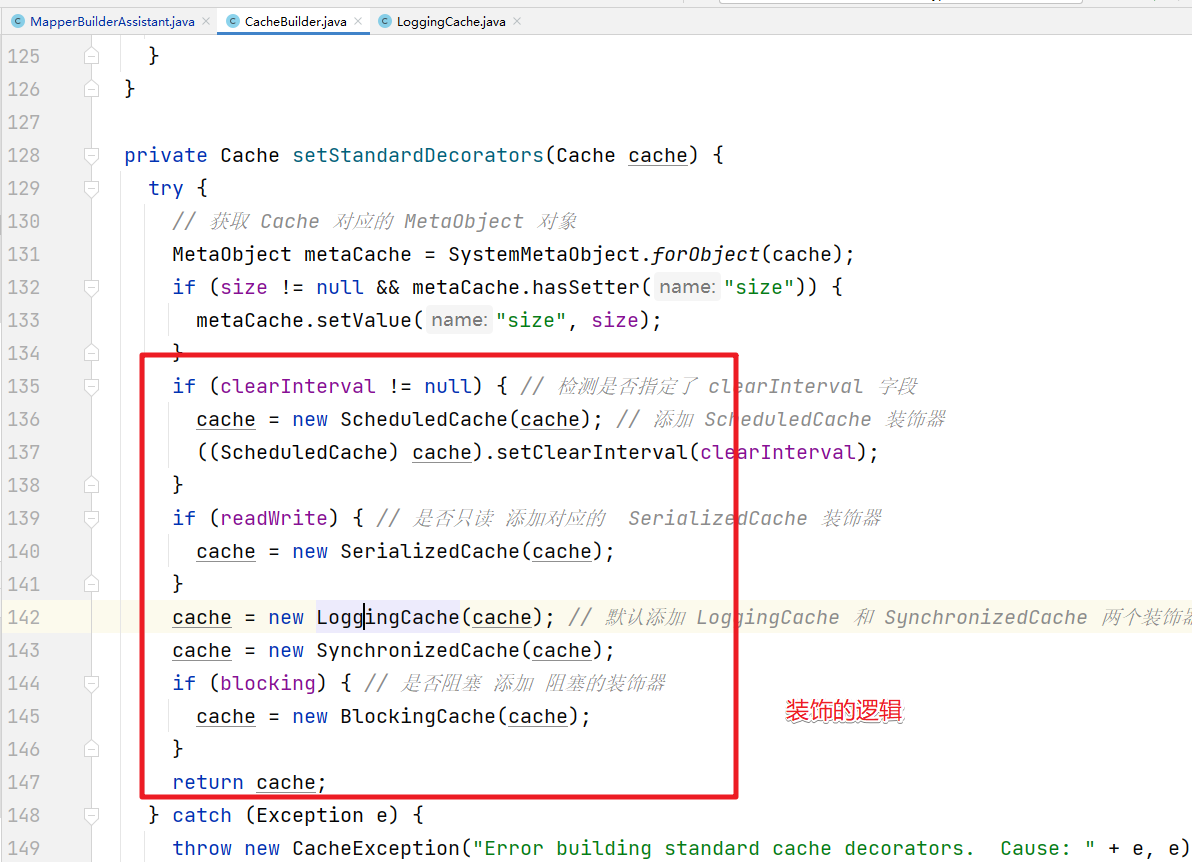
image.png

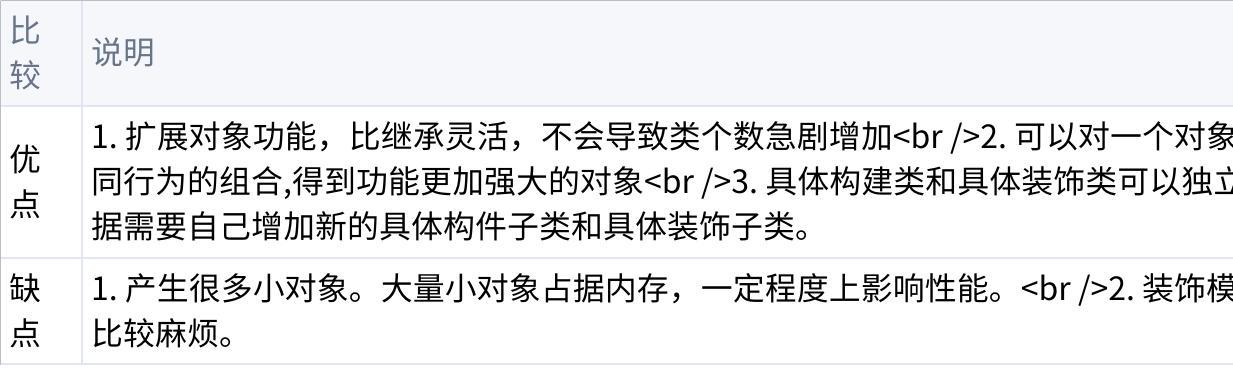
每个装饰类都有自己的作用

* BlockingCache：阻塞的
* LruCache:根据Lru规则来淘汰缓存数据
* FifoCache：根据FIFO规则来淘汰缓存数据
* …

源码中的装饰：







**点击图片可查看完整电子表格**

**行为型模式**

**迭代器模式**

**迭代器模式**是一种行为设计模式， 让你能在不暴露集合底层表现形式 （列表、 栈和树等） 的情况下遍历集合中所有的元素。