**SQL映射框架MyBatis源码分析**

主讲：Cat 老师

北京动力节点教育科技有限公司  
2020 • 北京

动力节点•版权所有•禁止传播

## 课程目标

1、该课程不会讲解mybatis框架如何使用，该框架应该是比较大众化的了，没有使用过该框架的学员应该是很少很少的；

2、该课程主要是分析mybatis框架的源码、底层实现 以及 手写自己的MyBatis框架；

## 阅读源码注意事项

1、源码看第一遍基本上都会晕车的，特别是之前没有看过源码的学员；

2、一开始看源码晕车很正常，建议至少看3遍以上，然后可能才感觉不晕车；

3、看源码目的主要了解其底层实现，设计思想，拓展思维，开发时借鉴和参考；

4、不必把每一个细节都弄明白，当今社会追求高效，不必重复造轮子；

5、看源码不必有完美主义，开源代码中也有不尽人意之处；

6、看源码主要掌握核心底层实现及精华之处，在工作中遇到异常能快速定位；

## 阅读源码指导原则

1、寻找入口

2、断点跟踪

3、先粗后细

4、精略结合

5、猜想验证

## 源码分析

1、建议先自己看一遍源码，然后再看直播或者录播视频，效果会更好；

2、建议看一遍第一期的Mybatis源码分析，可以把视频播放速度加到1.3-1.5-2.0倍播放，可以一天完成MyBatis所有视频的学习；

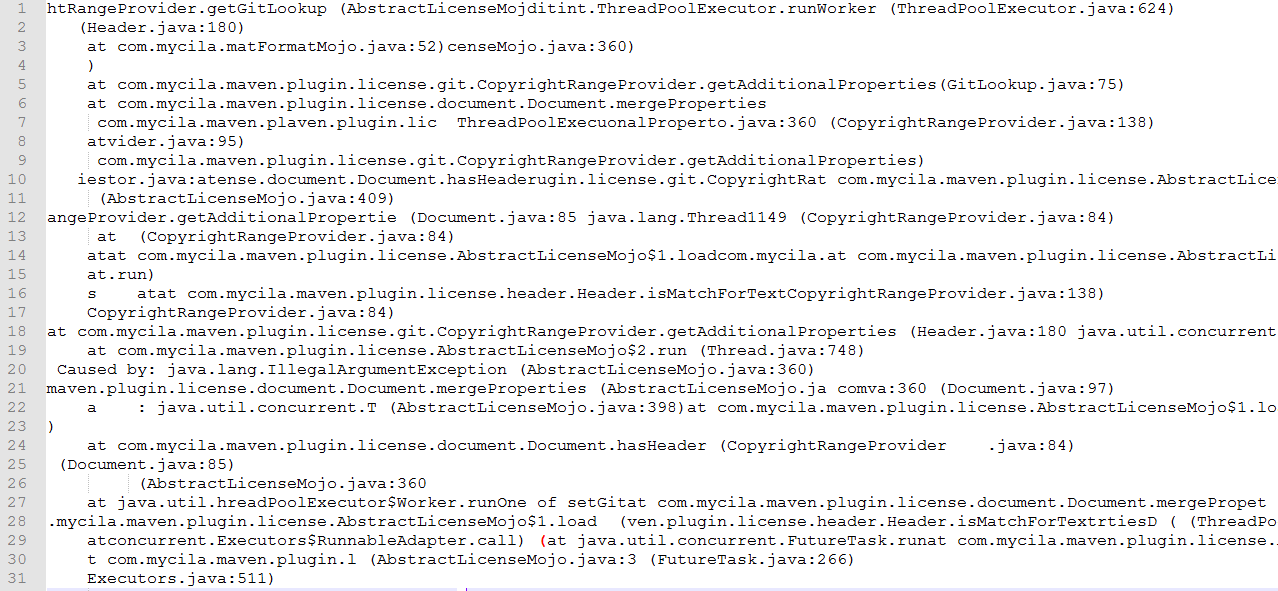
3、官方网站：<https://blog.mybatis.org/>

4、官方文档：<https://mybatis.org/mybatis-3/>

5、源码下载：<https://github.com/mybatis>

6、源码构建

从Github下载下来的源码导入到IDEA后，通过maven compile，会产生如下错误，将代码中的license.txt文件删除即可



1. IDEA安装Statistic插件，可以统计项目代码行数；
2. 带中文注释版本的MyBatis源码共享在Gogs仓库上的：

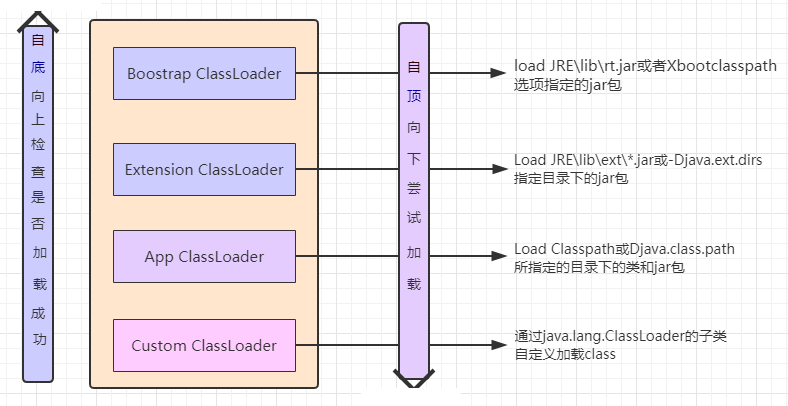
<http://114.116.213.93:3000/project/cat-space>

## 类加载器Classloader

ClassLoader翻译过来就是类加载器，一般开发时其实用得不多，但对于某些框架开发来说却非常常见；

ClassLoader的具体作用就是将class文件加载到jvm虚拟机中去，程序就可以正确运行了，但是jvm启动的时候，并不会一次性加载所有的class文件，而是根据需要动态加载，如果一次性加载那么多jar包那么多class，那内存消耗极大；

Java语言自带有三个类加载器：



**Bootstrap ClassLoader** 最顶层的加载类，主要加载核心类库，%JRE\_HOME%\lib下的rt.jar、resources.jar、charsets.jar和class等;

Bootstrap ClassLoader是C语言实现的；

**Extention ClassLoader** 扩展的类加载器，加载目录%JRE\_HOME%\lib\ext目录下的jar包和class文件；

**Appclass Loader**也称为SystemAppClass 加载当前应用的classpath下的所有类；

那么这3个类加载器的加载顺序是怎么样的，具体哪个先行？顺序是：

1. Bootstrap CLassloder
2. Extention ClassLoader
3. AppClassLoader

我们自己写的类是由AppClassLoader加载，jdk api提供的那些类，比如String，Object，Date是由Bootstrap CLassloder加载；

System.***out***.println(Test.**class**.getClassLoader());  
System.***out***.println(String.**class**.getClassLoader());

每个类加载器都有一个父加载器，比如加载Test.class是由AppClassLoader完成，那么AppClassLoader也有一个父加载器，通过getParent方法获取；

ClassLoader cl = Test.**class**.getClassLoader();  
System.***out***.println(**"ClassLoader is:"**+cl.toString());  
System.***out***.println(**"ClassLoader's parent is:"**+cl.getParent());  
System.***out***.println(**"ClassLoader's parent is:"**+cl.getParent().getParent());

ExtClassLoader的父加载器是null（因为是C语言实现的，所以打不出名字）；

## 纯MyBatis代码开发步骤

第一步：读取mybatis-config.xml配置文件；

第二步：构建SqlSessionFactory；

第三步：打开SqlSession；

第四步：获取Mapper接口对象；

第五步：调用Mapper接口对象的方法操作数据库；

## 具体代码实现：

*//第一步：读取mybatis-config.xml配置文件*InputStream inputStream = Resources.*getResourceAsStream*(**"mybatis-config.xml"**);  
  
*//第二步：构建SqlSessionFactory*SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(inputStream);  
  
*//第三步：打开SqlSession*SqlSession session = sqlSessionFactory.openSession();  
  
*//第四步：获取Mapper接口对象*UUserInfoMapper uUserInfoMapper = session.getMapper(UUserInfoMapper.**class**);  
  
*//第五步：调用Mapper接口对象的方法操作数据库；*UUserInfo uUserInfo = uUserInfoMapper.selectByPrimaryKey(1);  
  
 *//第六步：业务处理****log***.info(**"查询结果: "** + uUserInfo.getId() + **"--"** + uUserInfo.getPhone());

## 源码分析

**第一步：**

*//第一步：读取mybatis-config.xml配置文件*InputStream inputStream = Resources.*getResourceAsStream*(**"mybatis-config.xml"**);

通过Resources工具类，调用ClassLoader读取classpath下的**mybatis-config.xml**配置文件，得到一个输入流；

**第二步：（有关配置文件的解析与处理都在该步骤中）**

*//第二步：构建SqlSessionFactory*SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(inputStream);

1、创建一个SqlSessionFactoryBuilder对象：

**new** SqlSessionFactoryBuilder()；

2、调用SqlSessionFactoryBuilder对象的build方法：

build(InputStream inputStream)

3、创建XMLConfigBuilder 对象：

XMLConfigBuilder parser = **new** XMLConfigBuilder(inputStream, environment, properties);

4、创建XMLMapperEntityResolver对象：

**new** XMLMapperEntityResolver()

1. 创建XPathParser对象：

**new** XPathParser(inputStream, **true**, props, **new** XMLMapperEntityResolver())

1. 创建XPath对象：

XPathFactory factory = XPathFactory.*newInstance*();  
**this**.**xpath** = factory.newXPath();

1. 创建Document对象：（DOM解析）

*//JDK提供的文档解析工厂对象*DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();

*//创建一个DocumentBuilder对象*DocumentBuilder builder = factory.newDocumentBuilder();

*//解析输入源的xml数据为一个Document文件***return** builder.parse(inputSource);

8、创建Configuration对象

**new** Configuration()

注册类型别名

初始化Configuration对象的一些成员变量的默认值

9、通过XPath方法解析mybatis-config.xml文件：

parseConfiguration(**parser**.evalNode(**"/configuration"**));

将解析出来的信息封装到Configuration对象中；

10、通过持有Configuration对象创建一个DefaultSqlSessionFactory对象：

**new** DefaultSqlSessionFactory(configuration);

**总结：**

MyBatis解析配置文件的本质就是为了获得Configuration对象；

Configuration 对象可以理解是mybatis的XML配置文件在程序中的化身，是MyBatis非常重要的一个对象，里面封装了MyBatis的整个配置信息；

以上即为MyBatis框架源码分析的上半部分；