mybatis 封装了持久层操作映射，是持久层框架，以制定Sql，存储过程，⾼级映射。通过 Mapper接⼝类与数据库操作做映射。

### ⼀ Mybatis中主要类：

XMLConfigBuilder.class

这个类主要是解析sqlMapConfig.xml和mapper.xml文件

Configuration.class

主要是Mybatis的初始化配置，拦截器，属性，环境数据源，全局别名，Mapper (类、包 路径、xml)

SqlSessionFactoryBuilder.class

⽤来构建SqlSessionFactory的 可选⽅式： factoryBuilder.build(Configuration.class); 通过⾃⼰创建的配置实例 factoryBuilder.build(inputStream); 通过⼀个输⼊流为 xml的⽂件数据。

SqlSessionFactory

⽤来获取 SqlSession 实例的⼯⼚类 实现参考(DefaultSqlSessionFactory.class)

SqlSession

获得⼀个数据库连接的会话，实现参考DefaultSqlSession.class SqlSessionFactory openSession ⽅法获得这个类的实例。 new DefaultSqlSession(configuration, executor, autoCommit);

Executor

它的实现⽤于执⾏与数据库操作的功能。 query, update 其中实现了缓存的使⽤⽅法。 BaseExecutor 基础实现(⼦类：BatchExecutor, ReuseExecutor, SimpleExecutor) 这个类 中使⽤会开启⼀级缓存 (⼀级缓存仅仅在当前回话中启作⽤)。 CachingExecutor ⼆级缓存，如果启⽤⼆级缓存，则⾸先从⼆级缓存中获得数据，如果数 据不存在则从⼀级缓存获取数据，最后从数据库读取数据

### **⼆ Mybatis Java创建SqlSessionFactory**

TransactionFactory transactionFactory = new JdbcTransactionFactory(); Environment environment = new Environment("test", transactionFactory, dataSource);

Configuration config = new Configuration(environment); config.addMapper(XxMapper.class);

config.setCacheEnabled(true);

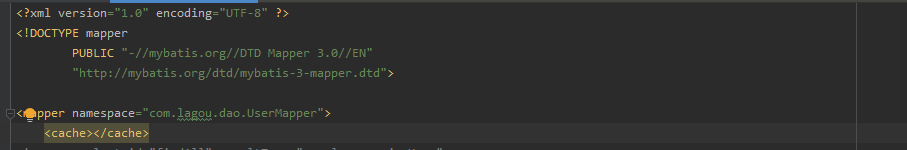
SqlSessionFactoryBuilder factoryBuilder = new SqlSessionFactoryBuilder();

SqlSessionFactory sqlSessionFactory = factoryBuilder.build(config);

### **三 二级缓存**

Mybatis默认开启二级缓存

<!--开启二级缓存  
 1、还需要在对应的mapper.xml中增加<cache></cache>  
-->  
<settings>  
 <setting name="cacheEnabled" value="true"/>  
</settings>



select 执⾏ SqlSession sqlSession = sqlSessionFactory.openSession(); XxMapper xxMapper = sqlSession.getMapper(XxMapper.class);

xxMapper.selectXxxx(…);

或

sqlSession.select(…);

XxMapper实际上通过Java动态代理的接⼝实现，调⽤sqlSession中的select,update等⽅ 法。 执⾏查询过程： 任何查询最终调⽤ selectList。 executor 实例(BaseExecutor⼦类 或 CachingExecutor) 默认启⽤⼆级缓存，则实例：CachingExecutor

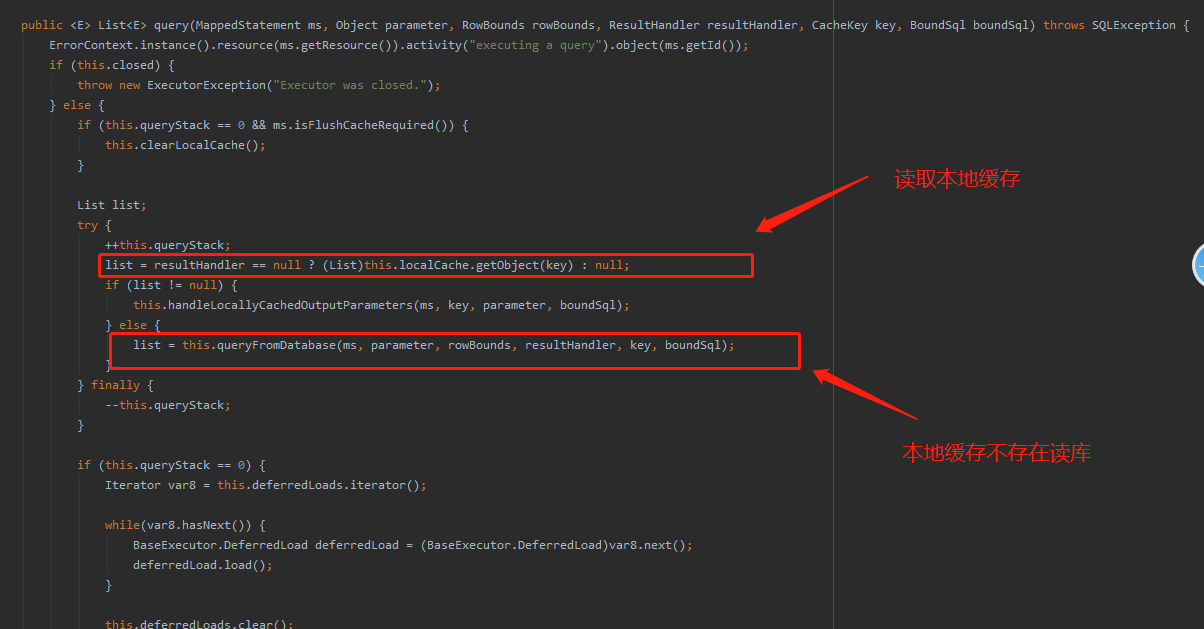
BoundSql 最终执⾏Sql等封装类，封装Sql语句与参数 CacheKey ⼀级、⼆级缓存的KEY，Mapper类名，⽅法名、执⾏sql等混合的Key。

创建好这些基本实例调⽤本类下 query(MappedStatement ms, Object parameterObject, RowBounds rowBounds, ResultHandler resultHandler, CacheKey key, BoundSql boundSql)⽅法，验证当前数据请求是否有⼆级缓存，如果没有则进⼊BaseExecutor.query

CachingExecutor.query如下图所示：



如果⼆级缓存实例存在，则进⼊缓存。 flushCacheIfRequired(ms); 刷新缓存，默认情况查询为false，则不会执⾏， ⾥⾯主要就是clear操作。



从上图可以看出，执⾏数据库读取操作之前，先从本地缓存获取值，如果本地缓存不存在， 则会从数据库取值，取出来值直接放到本地缓存中。因为每个SqlSession⼀个实例，且创建 的BaseExecutor单独⼀个实例，则localCache 仅在当前会话中有效。 localCache = PerpetualCache.class (这⾥可以⾃⼰翻阅代码看⼀眼) PerpetualCache⾥⾯ Map<Object, Object> cache = new HashMap<>();⽤来存的数据

⼆级缓存启动时：executor = CachingExecutor CachingExecutor.update 执⾏数据库修改插⼊语句。执⾏之前，会先将缓存标记为提交清 理，并且清空当前会话中select的提交的本地事务缓存数据。

从MyBatis 的update 过程中可以看出，任何修改都将会清理缓存数据，提交之前先清空本 地缓存，commit 之后，将⼆级缓存清空， 本地缓存清空。 select 的时候会创建事务缓存entriesToAddOnCommit， 这个缓存在提交事务之后，会还原 到⼆级缓存与本地缓存之中。但只为在update之后到select的查询结果会被还原。

值得注意的是使用mybatis默认的PerpetualCache二级缓存。如果是分布式应用的话会出现脏读，这时候就可以使用redis等缓存数据库来实现二级缓存。

需要修改对应的缓存实现类为RedisCache.class

@CacheNamespace(implementation = RedisCache.class)//开启二级缓存  
public interface UserMapper {  
  
 @Results({

需要依赖的jar包为

<dependency>  
 <groupId>org.mybatis.caches</groupId>  
 <artifactId>mybatis-redis</artifactId>  
 <version>1.0.0-beta2</version>  
</dependency>