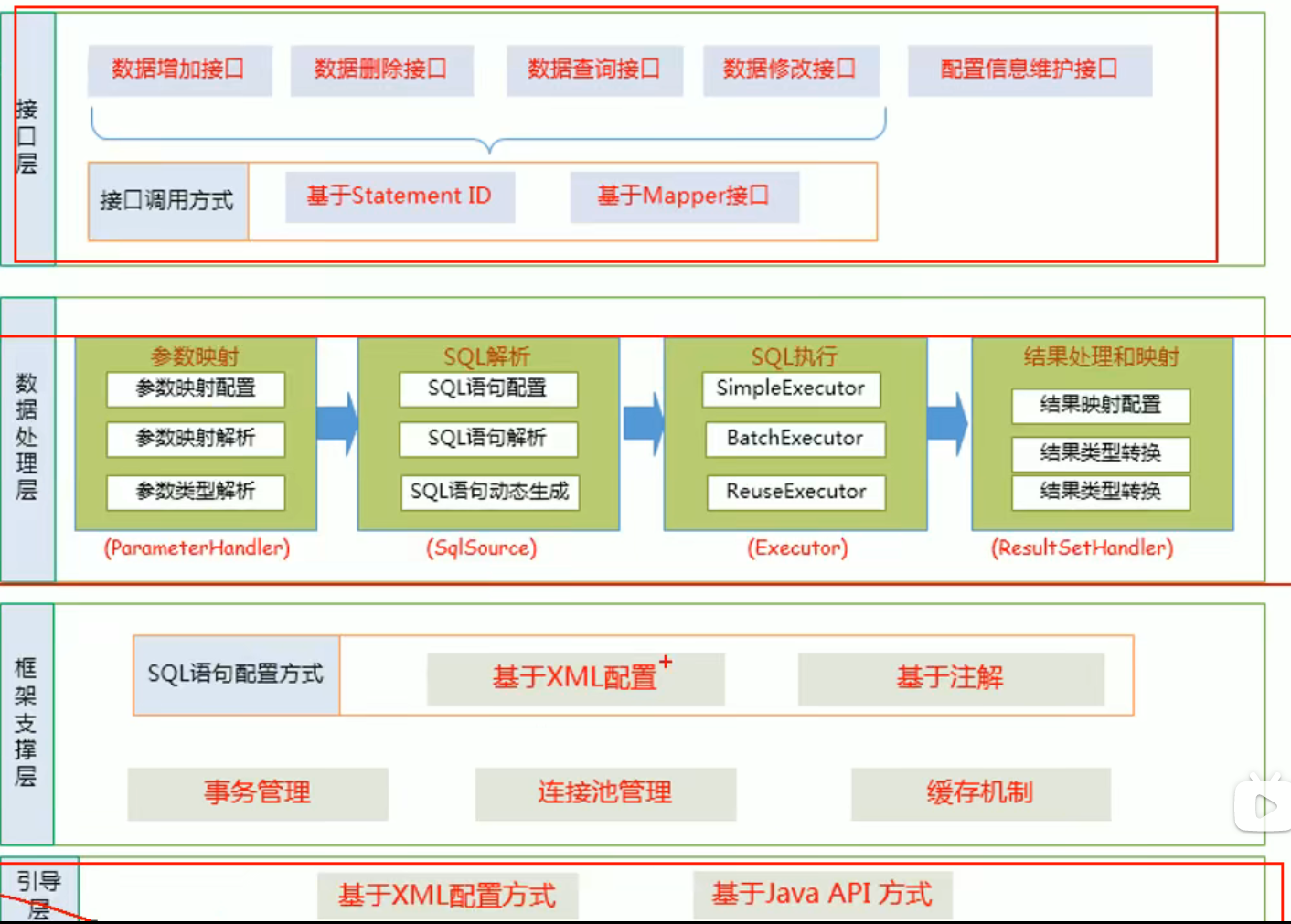
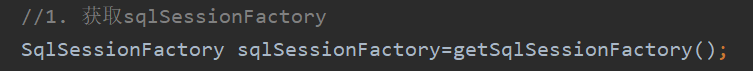
Mybatis的分层架构



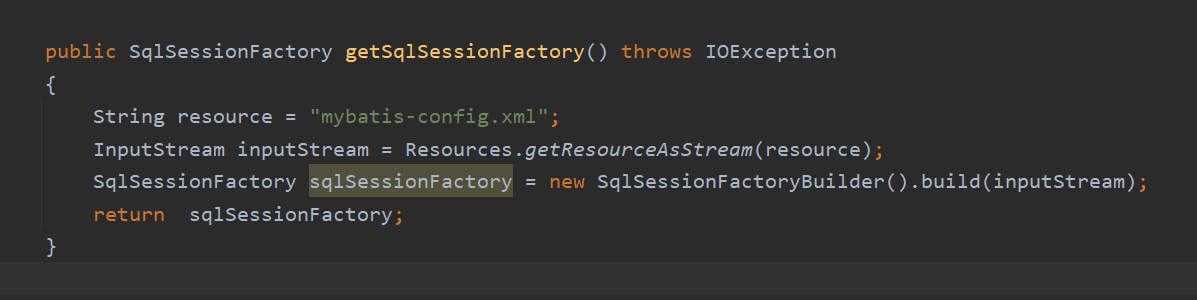
Myabtis的运行流程

1.获取SqlSessionFactory对象

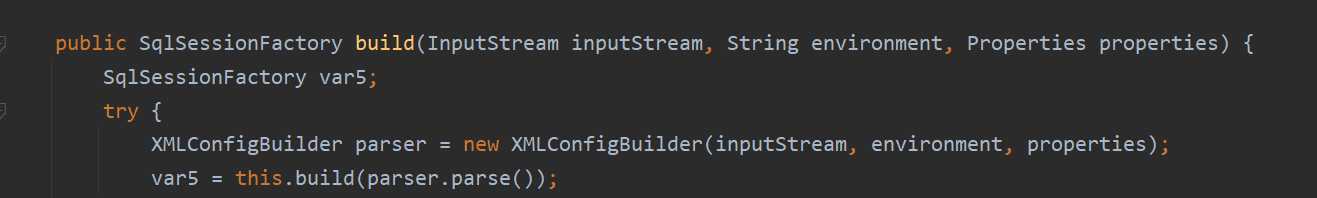


1)创建SqlSessionFactoryBuilder对象

2）将配置文件通过IO流的方式传给SqlSessionFactoryBuilder().build(inputStream)

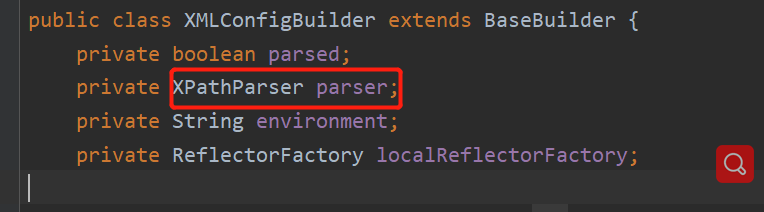


全局配置文件的xml

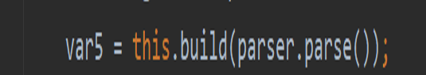


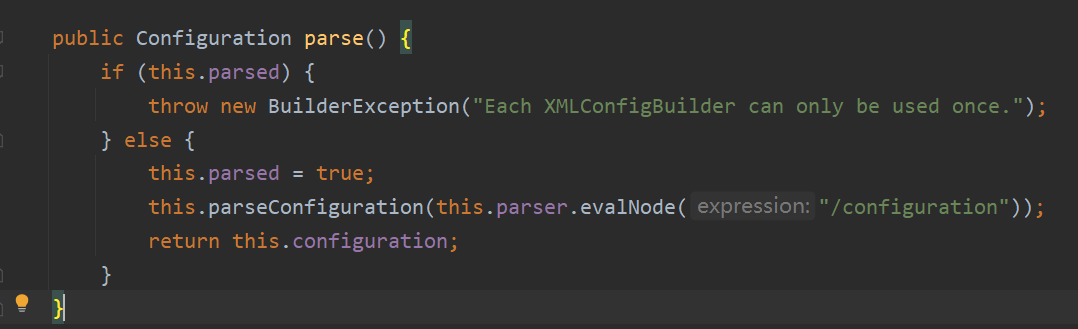
将Mybatis的全局配置文件的标签传入XMLConfigBuilder方法

3)创建 parser（解析器）



接着执行parser.parse()方法





因为该配置文件没被解析过

所以执行evalNode()，计算configuration节点，即全局配置文件中最大的标签

<configuration></configuration>

接着解析configuration

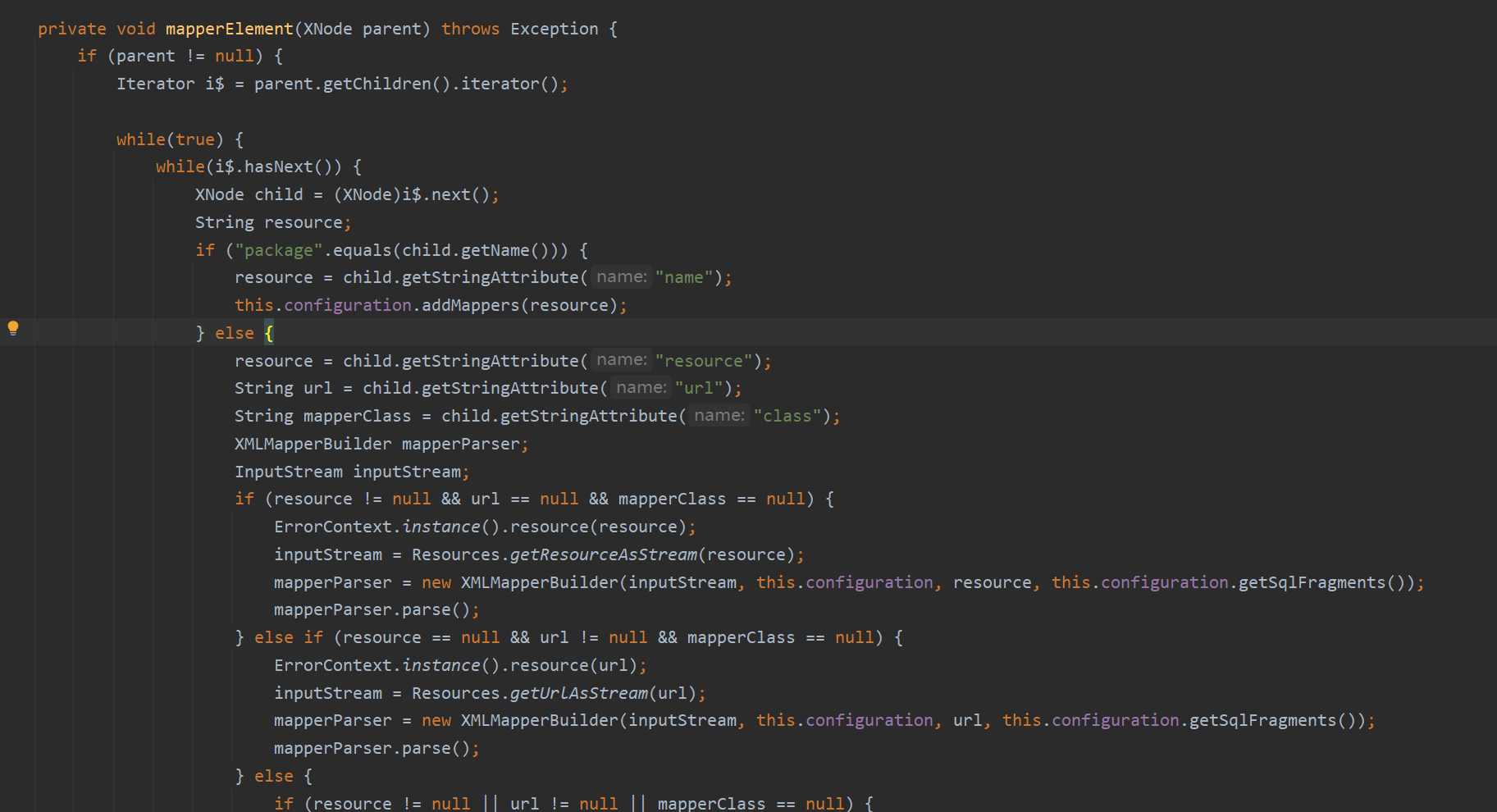


计算我们全局配置的每一项内容（这里以setting标签举例，比如是否开启懒加载，是否开启驼峰命名法）

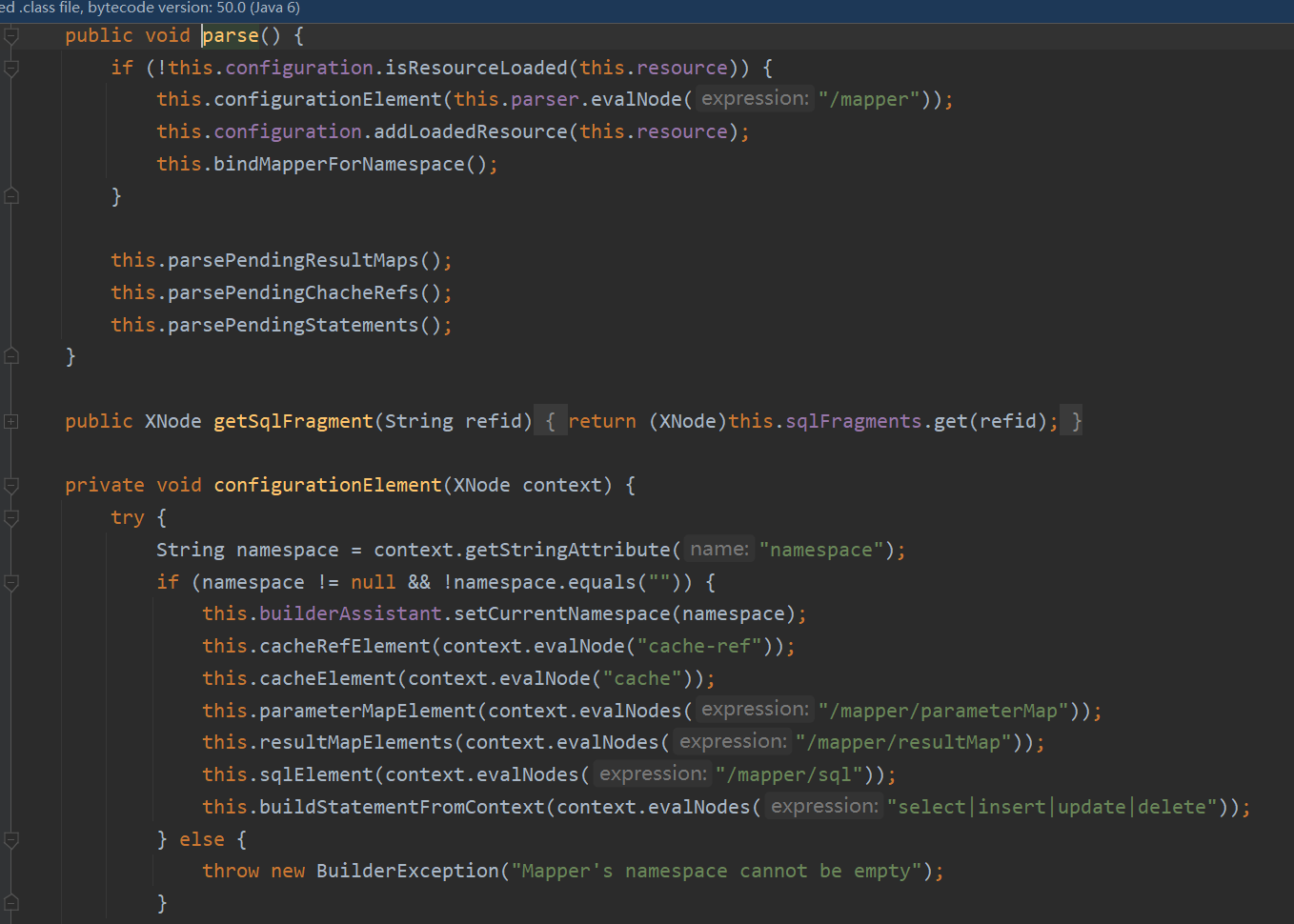


4)解析后产生一个Configuration对象,每一个标签保存在这个对象中

5)解析mapper.xml



Mybatis解析什么都是用一个解析器执行的



Mapper.xml中的每一个元素信息解析出来并保存在全局配置中将增删改查标签的每一个标签每一个属性都解析出来

封装成一个MapperStatement:代表一个增删改查标签的详细信息

然后再把这个MapperStatement装回给Configuration

MapperStatement是Configuration的一个重要属性

所以Configuration对象保存了所有配置文件的详细信息

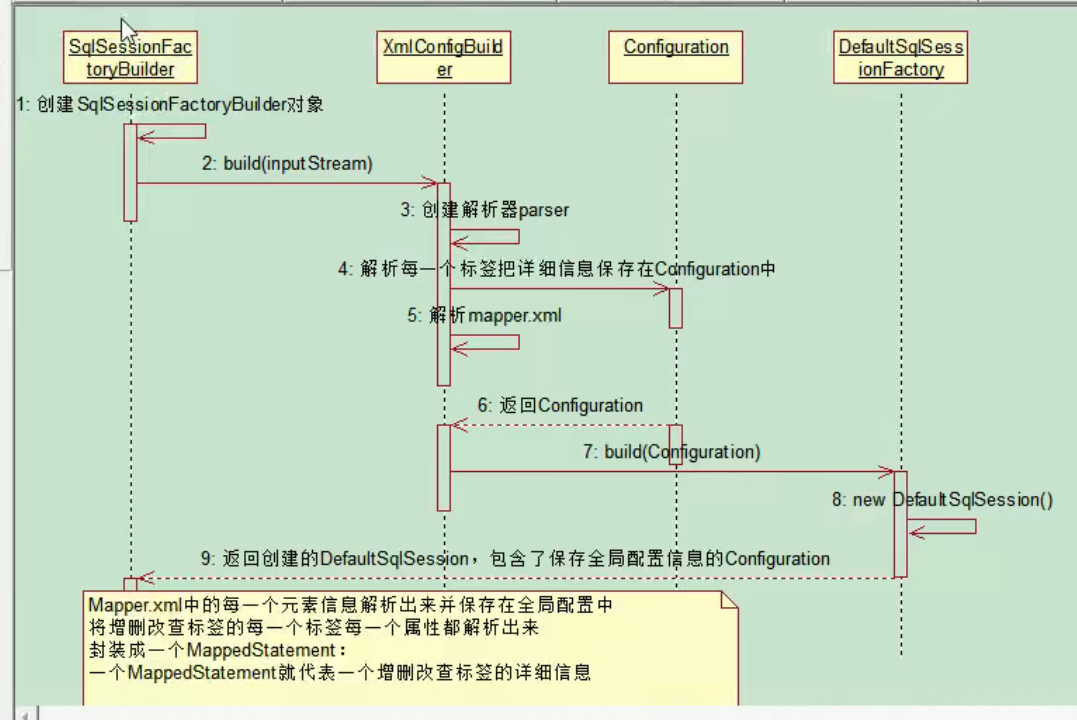
6）最终parse方法返回封装了所有配置信息的Configuration

7）build(Configuration)

8) new DefaultSqlSession()

9)返回创建的DefaultSqlSession,包含了保存全局配置信息的Confugiration

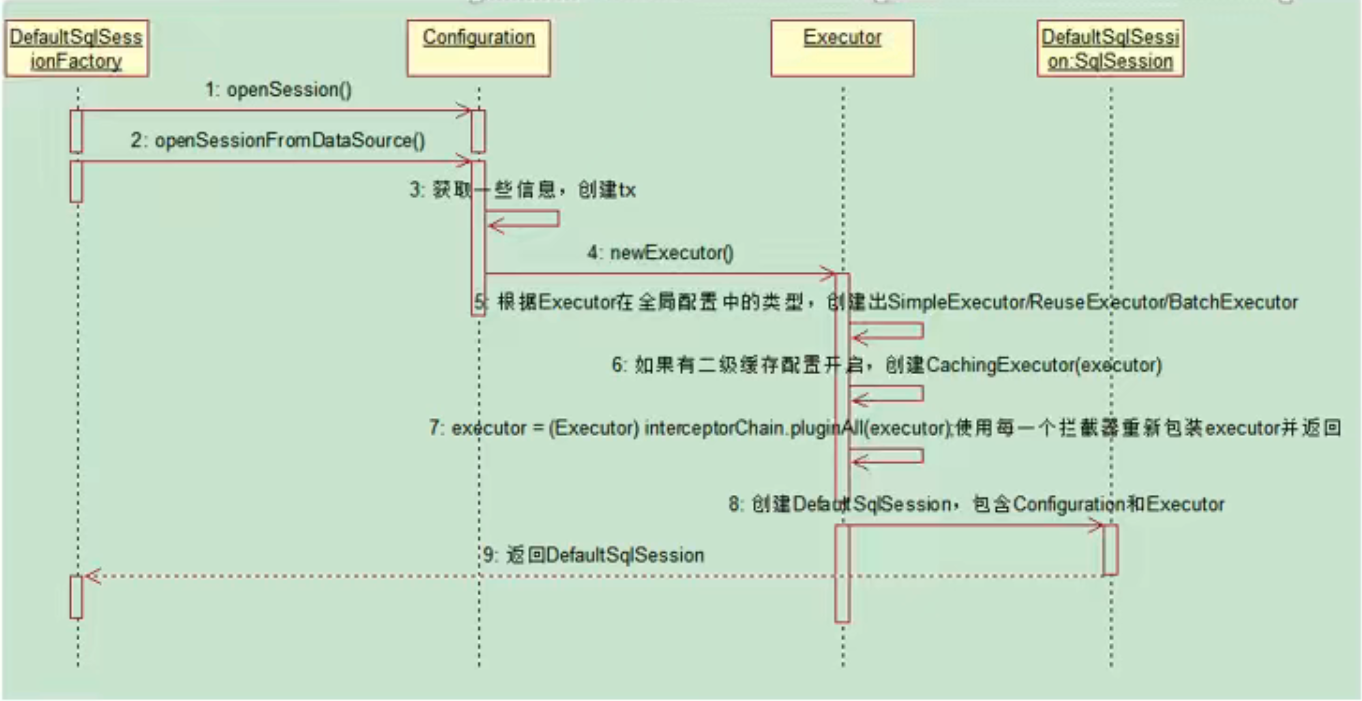
根据配置文件创建SQLSessionFactory



总结：把配置文件的信息解析并保存在Configuration对象中，返回包含了Configuration对象中，返回包含了Configuration的DefaultSqlSession

2.获取SqlSession对象

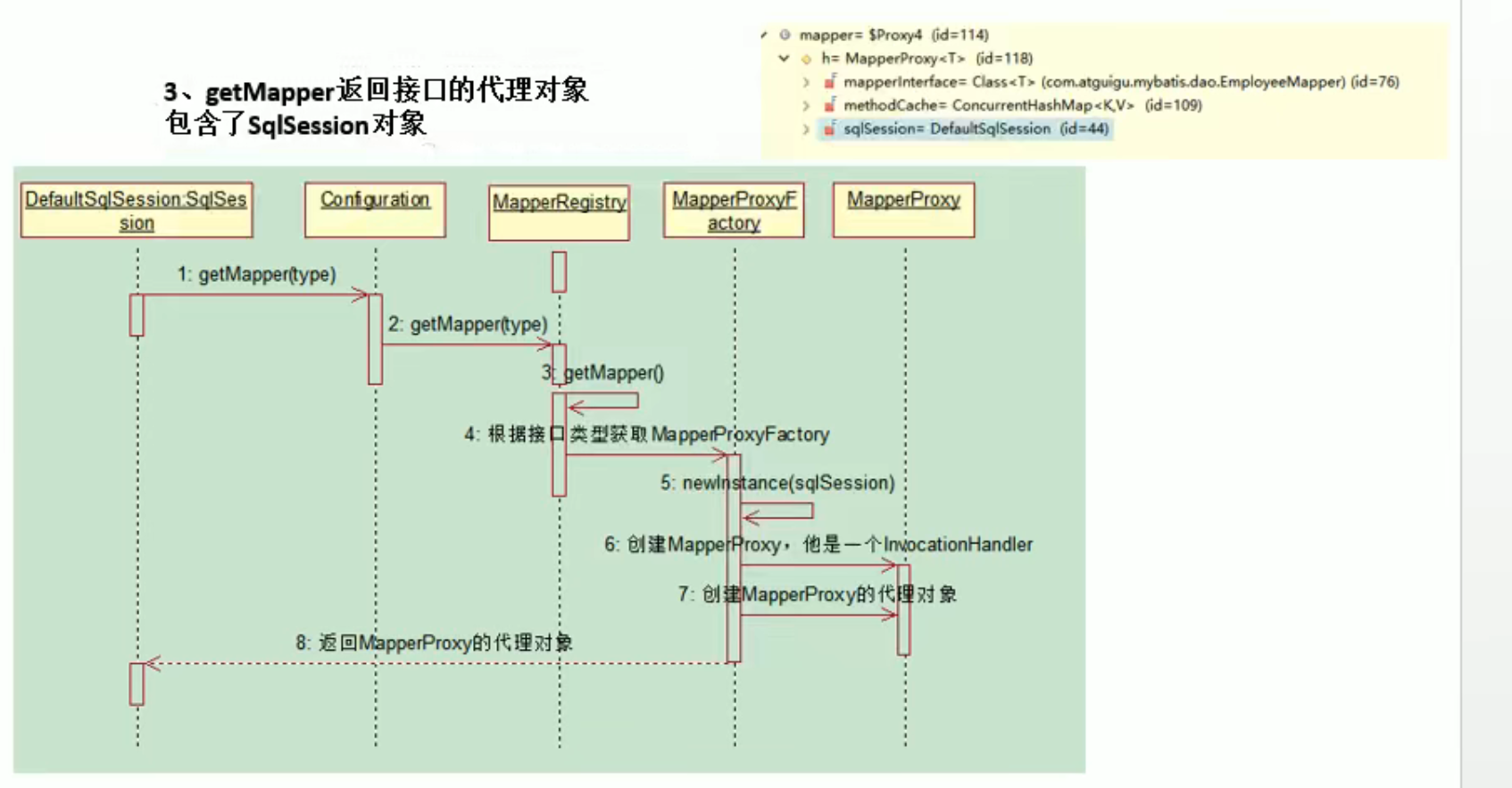
返回SqlSession的实现类DefaultSqlSession对象，他里面包含了Executor（用来做增删改查的，里面有update方法，query方法等）和Configuration，四大对象之一的Executor（执行器）在这一步被创建



默认是Simple

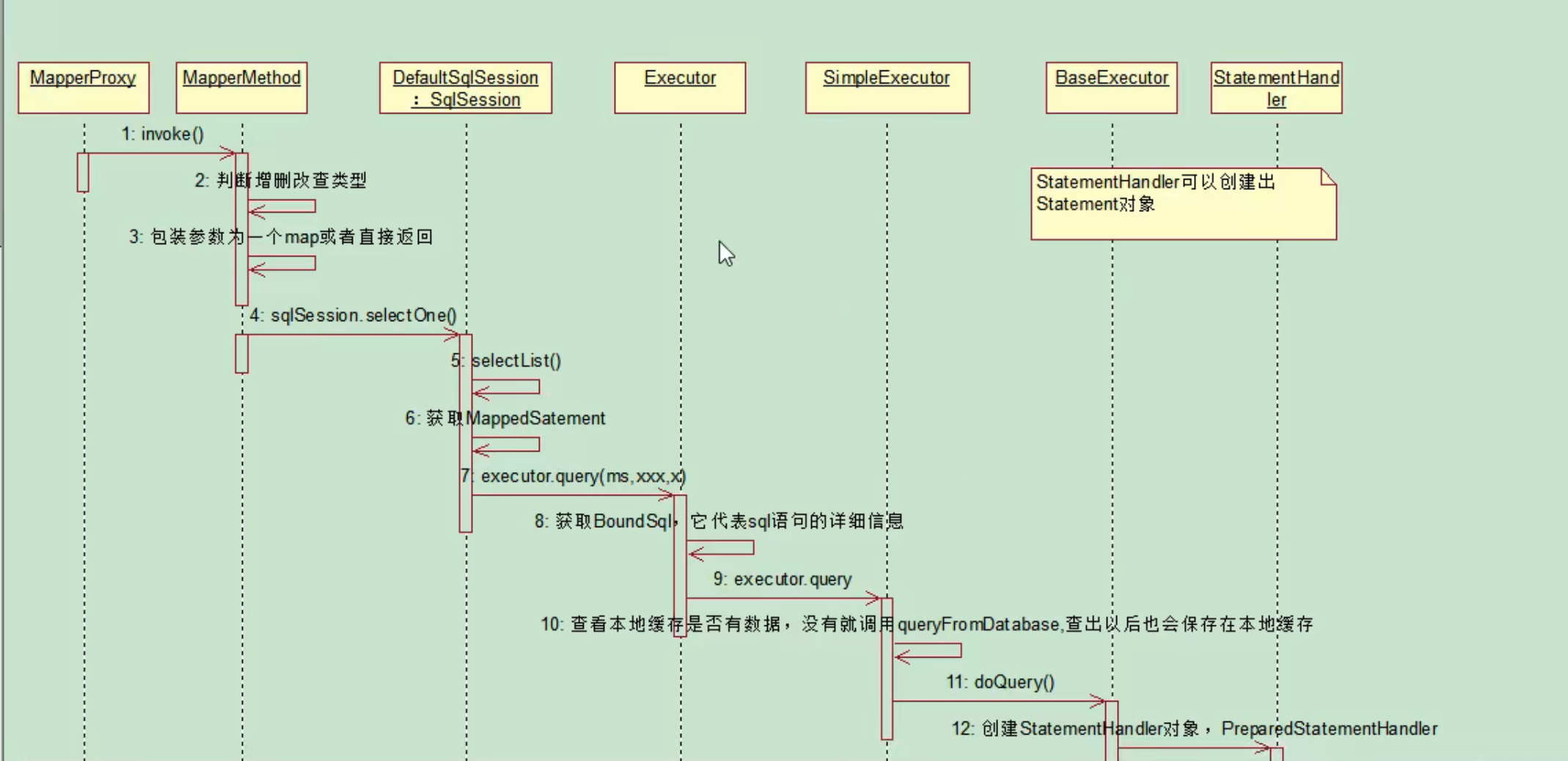
事务

3.getMapper获取到mapper接口的代理对象



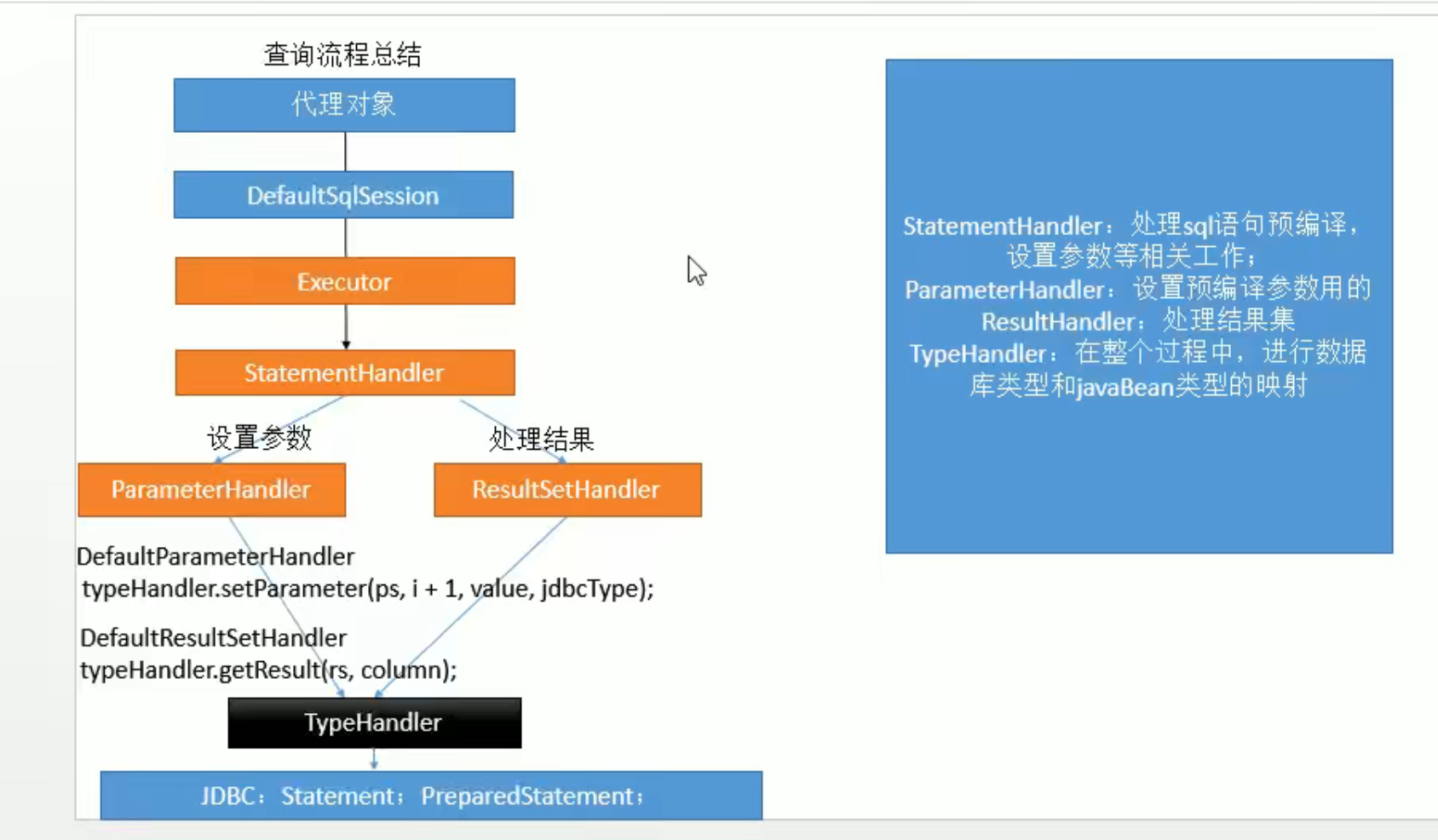
4.执行增删改查方法

查询实现：





查询流程总结：



5.总结

1.根据配置文件(全局配置文件，mapper（映射）文件)初始化出Configuration对象

2.创建一个DefaultSqlSession，里面包含了Configuration以及Executor（根据全局配置文件中的defaultExecutorType创建出对应的Executor

3.DefaultSqlSession.getMapper(),拿到Mapper接口对应的MapperProxy

4.MapperProxy里面有DefaultSqlSession

5.执行增删改查方法

1)调用DefaultSqlSession的增删改查(Executor)

2)创建StatementHandler对 象（同时也会创建ParameterHandler和ResultSetHandler）

3）调用StatementHandler的参数预编译

4）调用StatementHandler的增删改查方法

5）使用ResultSetHandler封装结果

注意：四大对象创建的时候都会有一个intercaptorChain.pluginAll(parameterHandler);