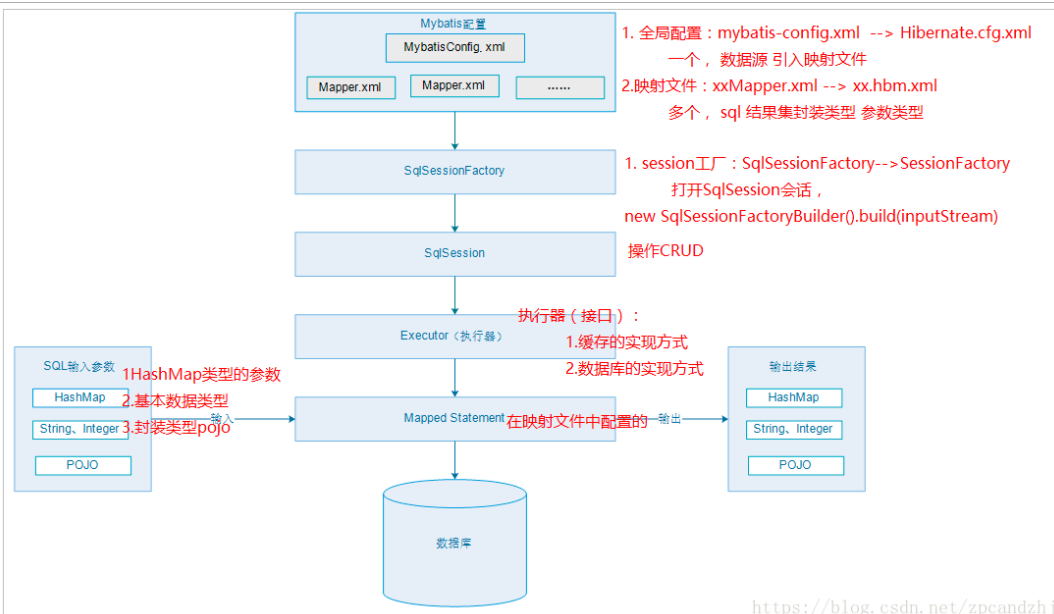
执行流程

1. 读取mybatis配置文件：mybatis-config.xml；
2. 加载映射文件，Mapper.xml
3. 构造会话工厂，SQLSessionFactory
4. 创建会话对象，SQLSession
5. Executor执行器
6. MappedStatement对象
7. 输入参数映射
8. 输出结果映射
9. 动态代理



**Mybatis缓存机制**

Mybatis有一级缓存和二级缓存。提升检索效率，避免每次查询都要查询数据库。

一级缓存：sqlsession级别缓存（本地缓存），每次执行sql都需要sqlsession，mybatis将查询出来的数据，放到本地缓存里，后续sql如果命中缓存的情况下，直接从缓存中读取数据。

二级缓存：跨sqlsession级别缓存，只要有sql拿到数据，就会将数据放到二级缓存中，其他sqlsession就可以从二级缓存中读取数据。

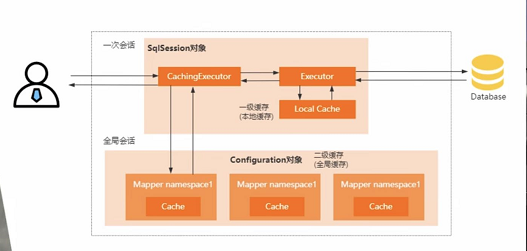
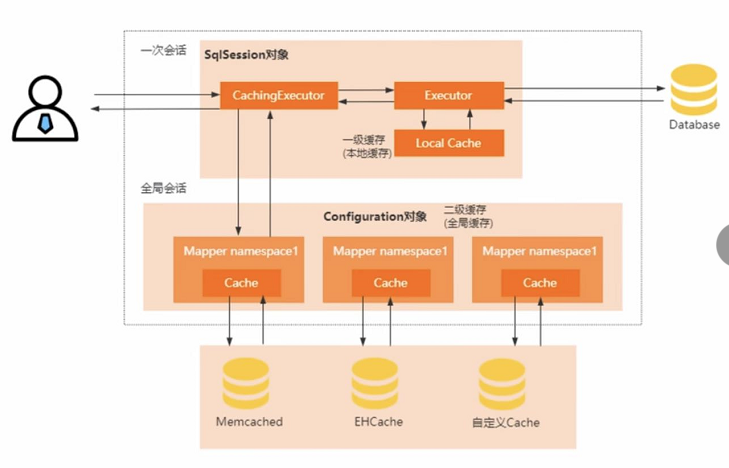
原理：

一级缓存，在sqlsession中会持有一个executor执行器，每个executor会持有local cache 对象，用户发起查询会从local cache中查找，没有缓存查询数据库。

二级缓存，会在executor上进行缓存的实现CacheExecutor 进行全局缓存，所以先查询二级缓存，再查询一级缓存，最后查询mysql。

生产上不建议使用缓存，避免脏读。

Mybatis的二级缓存是和命名空间绑定的，所以通常情况下每一个Mapper映射文件都有自己的二级缓存，不同的mapper的二级缓存互不影响。这样的设计一不注意就会引起脏读，从而导致数据一致性的问题。引起脏读的操作通常发生在多表关联操作中，比如在两个不同的mapper中都涉及到同一个表的增删改查操作，当其中一个mapper对这张表进行查询操作，此时另一个mapper进行了更新操作刷新缓存，然后第一个mapper又查询了一次，那么这次查询出的数据是脏数据。出现脏读的原因是他们的操作的缓存并不是同一个。



Executor执行器种类：

·SimpleExecutor，每次执行一次update或select，就开启一个statement，用完后关闭。

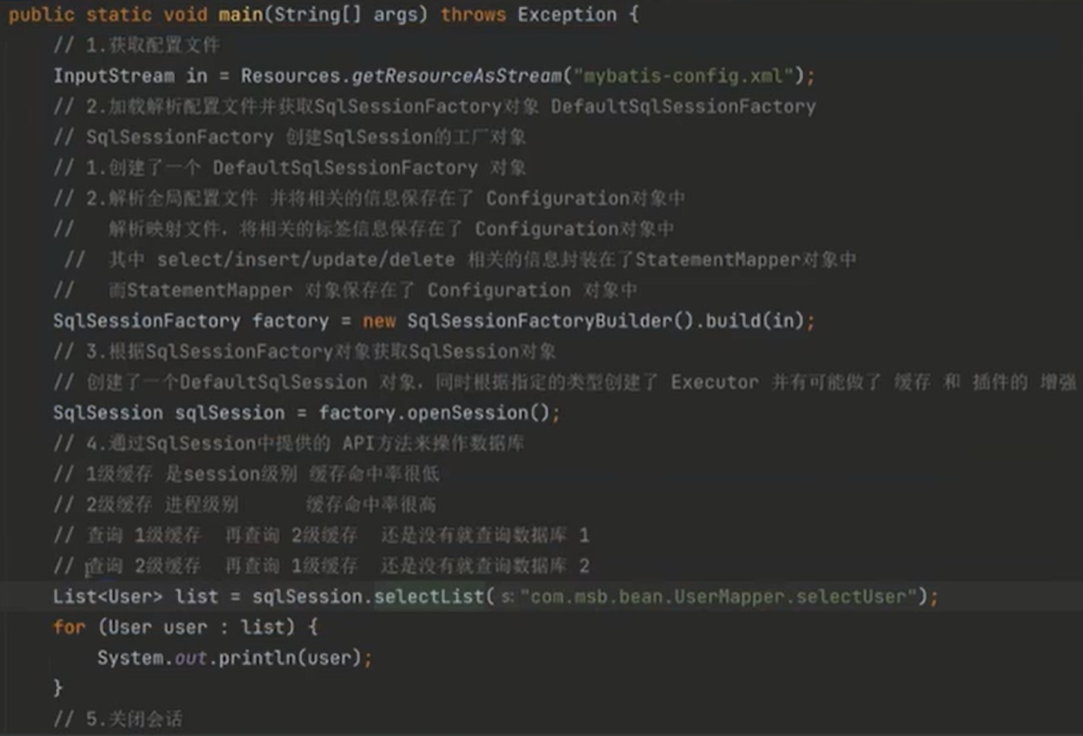
· ReuseExecutor，在执行update或select时以sql作为key去查询statement，有则直接使用，没有则创建，使用完不关闭，放入Map<String,Statement>中，供下次使用。减小开启statement的时间。

·BatchExecutor，执行update（jdbc不支持select），会把所有sql添加到批处理addBatch(),等待统一批处理executorBatch()；它缓存了多个statement，每一个statement都是addBatch()。

作用范围：都是sqlSession生命周期范围内。

http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd 文件作用

约束了在mybatis-generator.xml文件中能写的内容



Mybatis用到的设计模式？

·缓存模块，装饰器模式

·日志模块，适配器模式、代理模式

·SQLSessionFactory,工厂模式

·Mapper接口，代理模式

·SQLSessionFactoryBuilder，建造者模式

DefaultSQLSession是线程安全的吗？

不是线程安全的。

Spring整合mybatis解决线程安全。

线程不安全的原因是多个线程操作同一个对象，而这个对象被定义为成员变量。

解决方案：将成员变量变成局部变量，让多个线程不能同时操作。

通过SqlSessionTemplate创建模板对象，定义了数据库操作的相关方法，本质上是通过代理对象，获取了DefaultSQLSession对象来执行，而且将DefaultSQLSession对象声明在局部变量中，从而解决线程不安全问题。