1:mybatis基础配置：首先是配置采用JDBC事务管理，数据库连接基础配置，其次是引入写有sql的映射文件（映射器）。再在映射文件中指明java代码所在的类，在此类中只需写与映射文件同名的接口即可调用。还要调用mybatis的核心组件：

SqlSessionFactoryBuilder（构造器）：根据配置信息或是代码来生成SqlSessionFactory

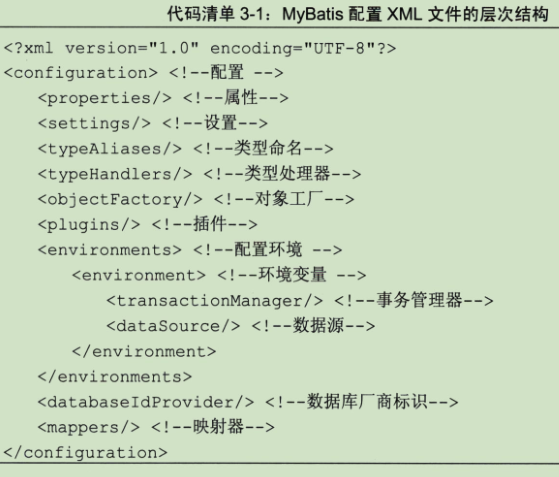
SqlSessionFactory：（工厂接口）：依靠工厂生产SqlSession；

SqlSession：一个既可以发送SQL去执行并返回结果，也可以获取mapper的接口；

2: SqlSessionFactoryBuilder的作用是获取资源来构建SqlSessionFactory，可构建多个；

SqlSessionFactory的作用是创建SqlSession，SqlSession相当于JDBC的connection，每次应用程序访问数据库就用通过SqlSessionFactory创建SqlSession，故SqlSessionFactory应存在mybatis整个生命周期，我们应该采用单例模式，一个数据库对应一个SqlSessionFactory，避免每次创建SqlSessionFactory都打开数据库连接避免连接资源被消耗。

SqlSession生命周期在请求数据库处理事务的过程中，是一个线程不安全的对象，设多线程操作数据库是就要注意器隔离级别，数据库锁等。每次用完要及时关闭，避免造成数据库连接池资源减少。



<property>:配置如数据库连接在内的属性值；

xml配置文件引入数据库配置文件：<propertice resource=”文件名”/>;

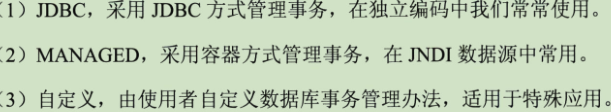
setting设置，关系到mybatis的主要设置，关系到mybatis运行时的行为方式。

<typeAliases>别名设置（有注释用法）。

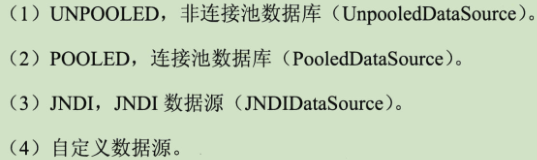
<typeHandlers>:类型转换例：java的byte对应JDBC的BYTE类型，一般系统自带的类型转换基本够用；xml配置文件中引入resultMap提供映射规则。枚举类型特殊处理：

C:\Users\Administrator\Desktop\捕获.PNG

Environment环境配置事务管理器：

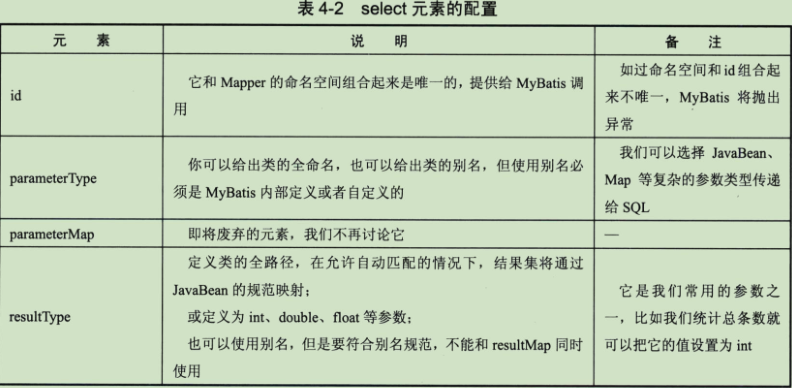
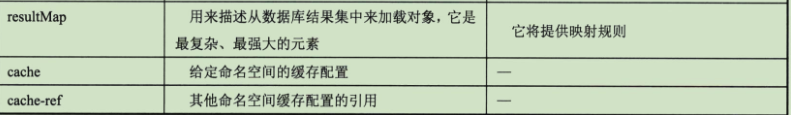
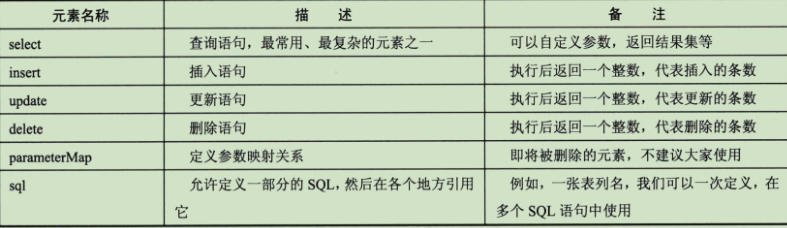


dataSource数据库连接方式



Mybatis可以配置数据库标签；

映射器元素：



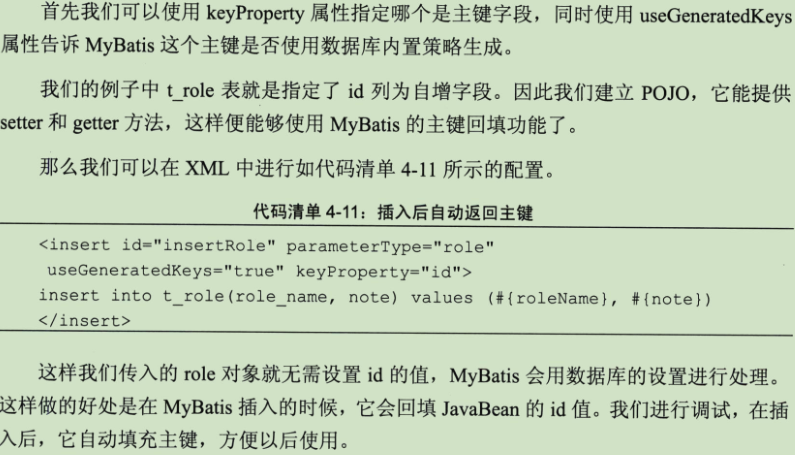
Mybatis的setting设置来配置全局属性如：使用别名和使用驼峰命名转换。

Select元素传递参数：parameterType，返回结果：resultMap；传递多个参数可以将parameterType设置为map或是类路径（即对象）来实现传多个参数，使用map的接口

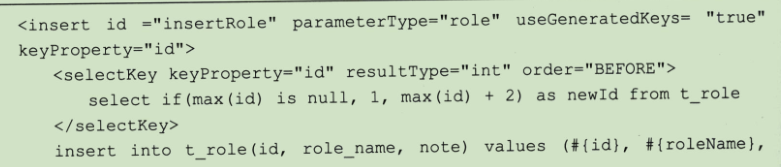
C:\Users\Administrator\Desktop\捕获.PNG

返回值用resultMap会自动根据结果的数量来判断是返回单个对象还是list

Mybatis设置自增主键：



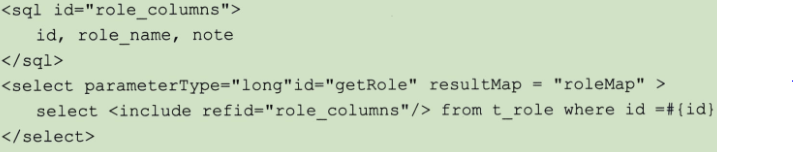
Id自增规则为最大id加2，无记录为1；

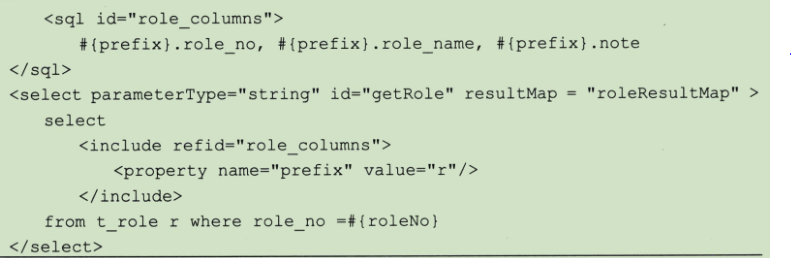


Insert，delete，update在执行完都会返回一个整数，标出执行后影响的记录条数；

Mybatis定义参数类型（即java对应于数据库的类型）的时候不允许换行

Sql元素：定义一串SQL语句的组成部分，其他语句可以通过引用来使用它；这样就可以实现SQL语句的复用（一处定义，多处使用）。也可以制定参数来使用它们。

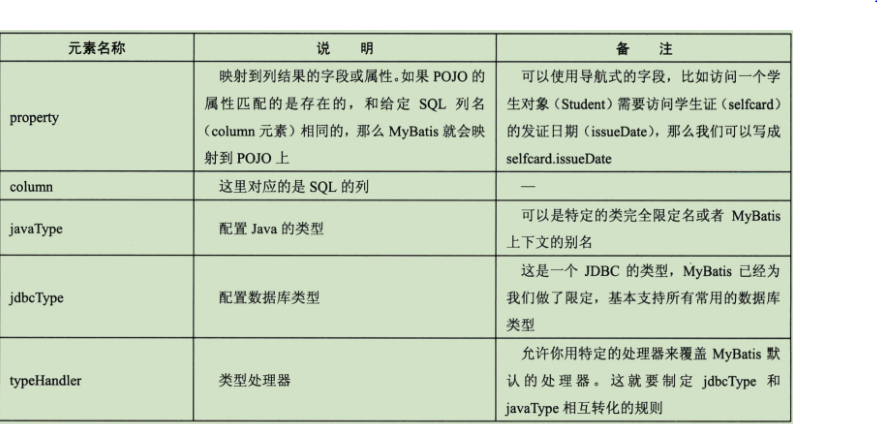




resultMap元素：其里面的元素：



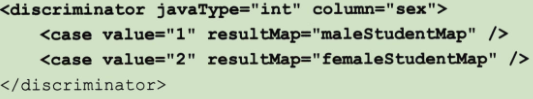
其中constructor元素用于配置构造方法，id元素表示那个列是主键，result是配置对象和SQL列名的映射关系，result和id的属性：



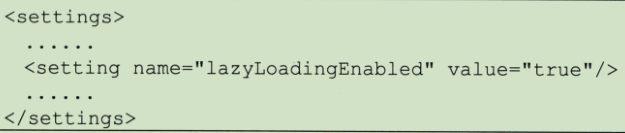
一般而言，任何select语句都可以使用map存储。使用map原则上可以匹配所有结果集，但可读下降，故一般用Java对象。

级联（a的查询结果作为b的查询条件）：一对多，多对一等关系，association代表一对一关系，collection代表一对多关系，

Discriminator是鉴别器（它可以根据实际选择采用那个类作为实力，允许根据特定的条件去关联不同的结果集），



延迟加载：一开始并不去除级联数据，只有当使用它时才发送SQL去取数据.开启延迟加载:



C:\Users\Administrator\Desktop\捕获.PNG

aggressiveLazyLoading设true则使用层级加载的策略，false为延迟加载；以上是全局修改

延迟加载可以对每个语句单独设置：在association和collection元素加入属性值fetchType就可以，有两个取值：eager和lazy（延迟加载）。

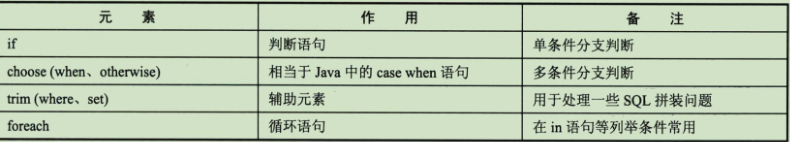
缓存：配置：

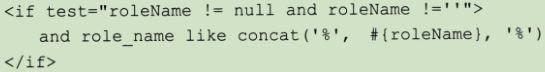
C:\Users\Administrator\Desktop\捕获.PNG

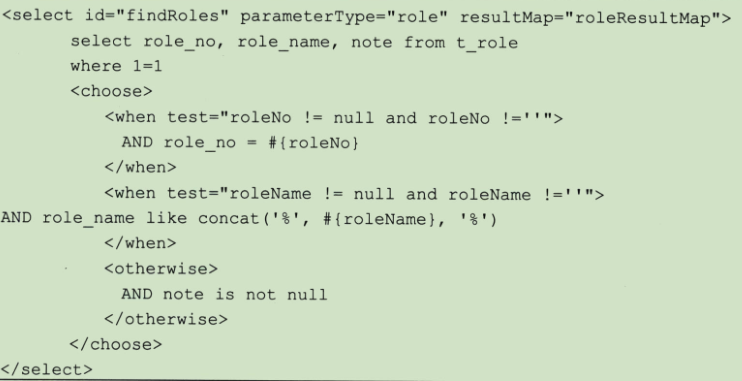
Select语句将会缓存，insert ，update，delete语句会刷新缓存

Eviction代表缓存回收策略，flushInterval刷新时间间隔单位毫秒，不设置在执行语句时刷新，size应用数目，代表缓存最多可以存储的对象个数，readonly缓存数据只能读不能改。

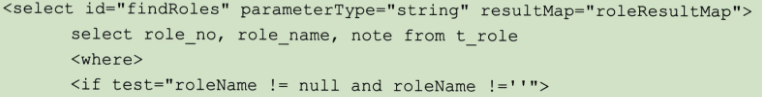
动态SQL：动态SQL元素：

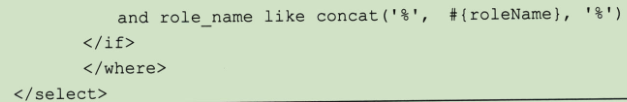




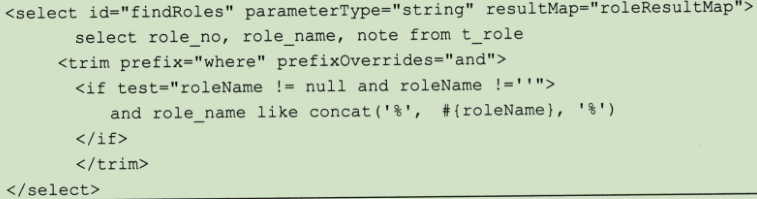


如果将1=1去掉则会变为错误SQL语句，但使用<where>元素可以避免（只有当where元素中的条件成立时才会组装到SQL语句中）：

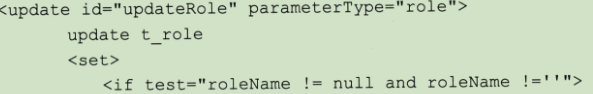


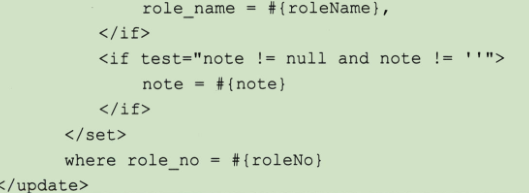


<trim>元素（意味着我们要去掉一些特殊的字符串，prefix：语句前缀，prefixOverrides：要去掉那种字符串）：

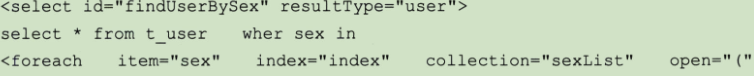


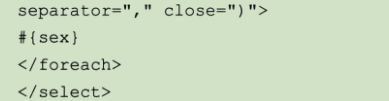
<set>:update用来更新某一个或是某个字段



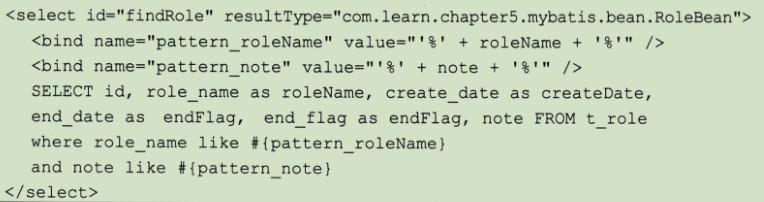
<test>:用于条件判断

<foreach>(collection：配置传递进来的参数名称，可以是一个数组或是list，item配置的是循环中当前元素。Index配置的是当前元素在集合的位置下标，open和close配置的是以什么符号将元素包装起来，separate是各个元素的间隔符):



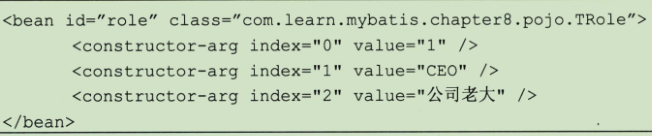
in语句性能消耗较大。

<bind>元素的作用是自定义一个上下文变量，用例：

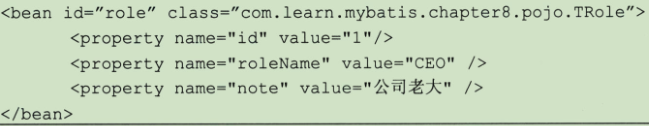


用此语句进行模糊查询，可用于Mysql和oracle（这两种的模糊查询SQL语句不同）.

Spring基础：构造器注入：



Setter注入：



接口注入（一种注入其他服务的接口）：

