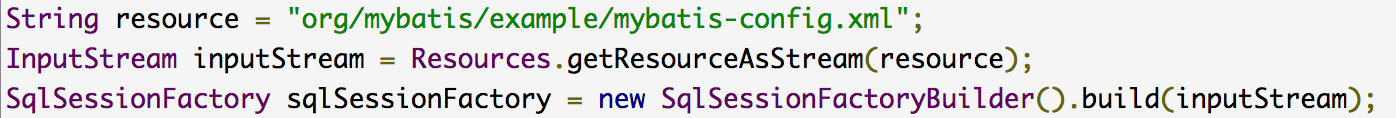
# 简介

Mybatis持久层框架，自定义SQL与映射，不需要处理JDCBC的设置参数与获取结果集的操作。

# 入门

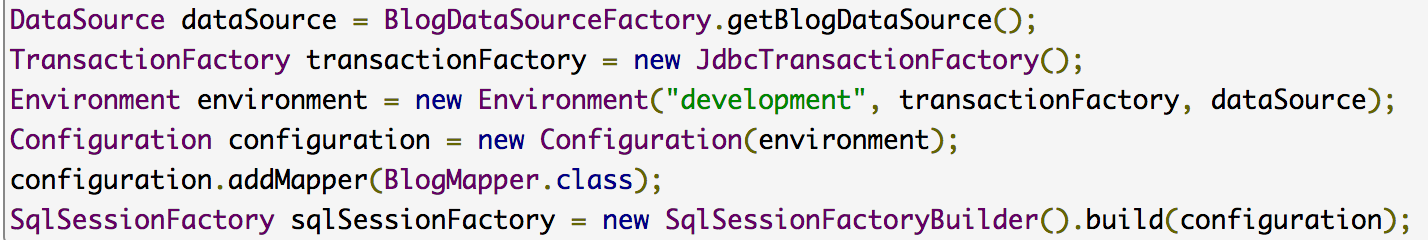
每个MyBatis应用都是以SqlSessionFactory为中心，SqlSessionFactoryBuilder通过XML配置文件或者Configuration实例构造Factory；XML配置文件的形式可以以任意（类路径或者文件路径）的方式被加载进来，Mybatis提供了Resources工具类；比如：



XML配置文件中包含Mybatis系统的核心配置，包含数据源配置，事务配置等。



程序的创建方法如下：



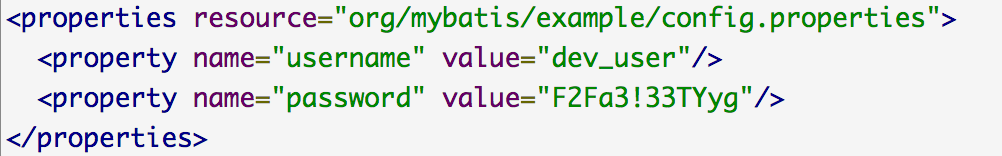
在不明确给出xml配置文件的情况下，Mybatis优先加载相同类路径下的同名XML文件加载，构建完SqlSessionFactory后，可以得到SqlSession对象来执行SQL命令；XML映射文件的命名空间与ID是与接口类相关的，这个命名可以直接映射到在命名空间中同名的 Mapper 类，并将已映射的 select 语句中的名字、参数和返回类型匹配成方法。这样你就可以像上面那样很容易地调用这个对应 Mapper 接口的方法，命名空间是必须的，完全限定名将被直接查找并且找到即用，SqlSessionFactoryBuilder只是用于构造SqlSessionFactory，构造完成后就没用了，作用域最好是方法作用域，SqlSessionFactory创建后要一直存在，作用域是应用作用域；SqlSession为线程作用域级别，通常是请求与方法作用域级别，它不是线程安全的，所以不能被共享，Mapper类映射器是用来绑定映射语句的接口，映射器接口的实例是通过SqlSession获取的，映射器的最大作用域与SqlSession相同，最佳作用域是方法作用域。

# XML映射配置文件

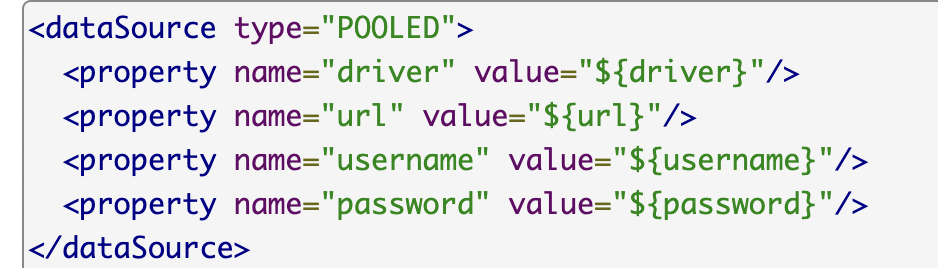
Mybatis的配置文件包含了设置与属性的信息。



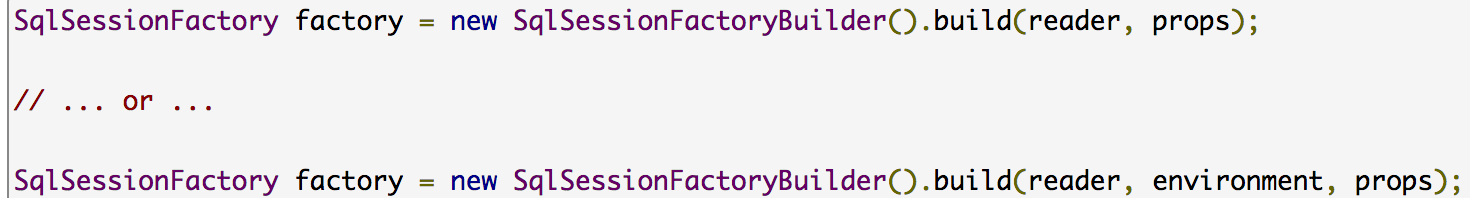
属性是外部配置而且可替换的，可以是外部的properties文件，也可以是property元素定义属性，比如：



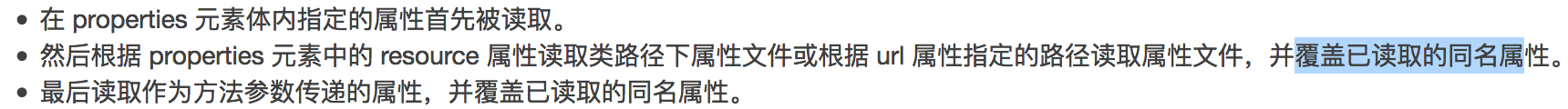
属性可以被用在配置文件中，比如：



属性也可能被传递到SqlSessionFactoryBuilder.build()中，比如：



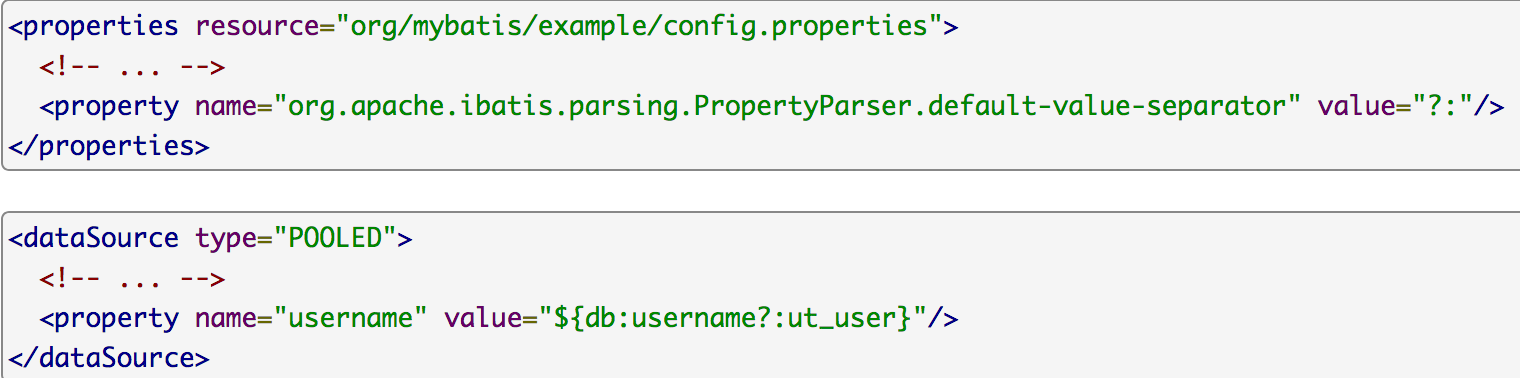
如果属性在多个地方配置，则加载顺序为：



也可以为占位符指定默认值，:号之后指定默认值，但是需要开启默认值的设置属性：



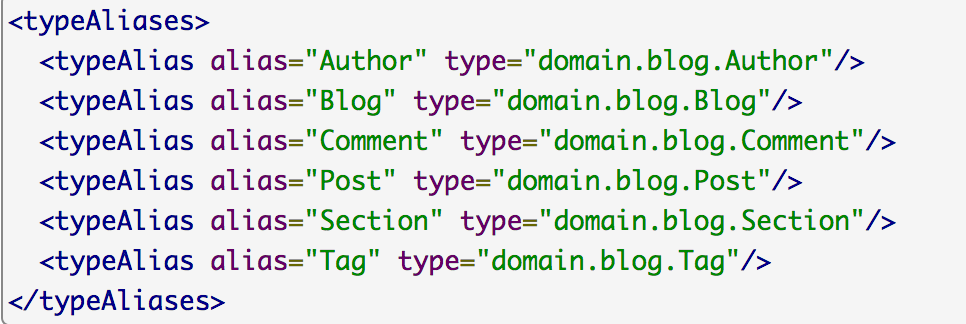
但是需要注意的一点是:符号也是属性分隔符的情况下，为了避免解析错误，需要重新定义默认值分隔符。



<settings>里面存储重要的设置，具体有哪些属性看参考文档。



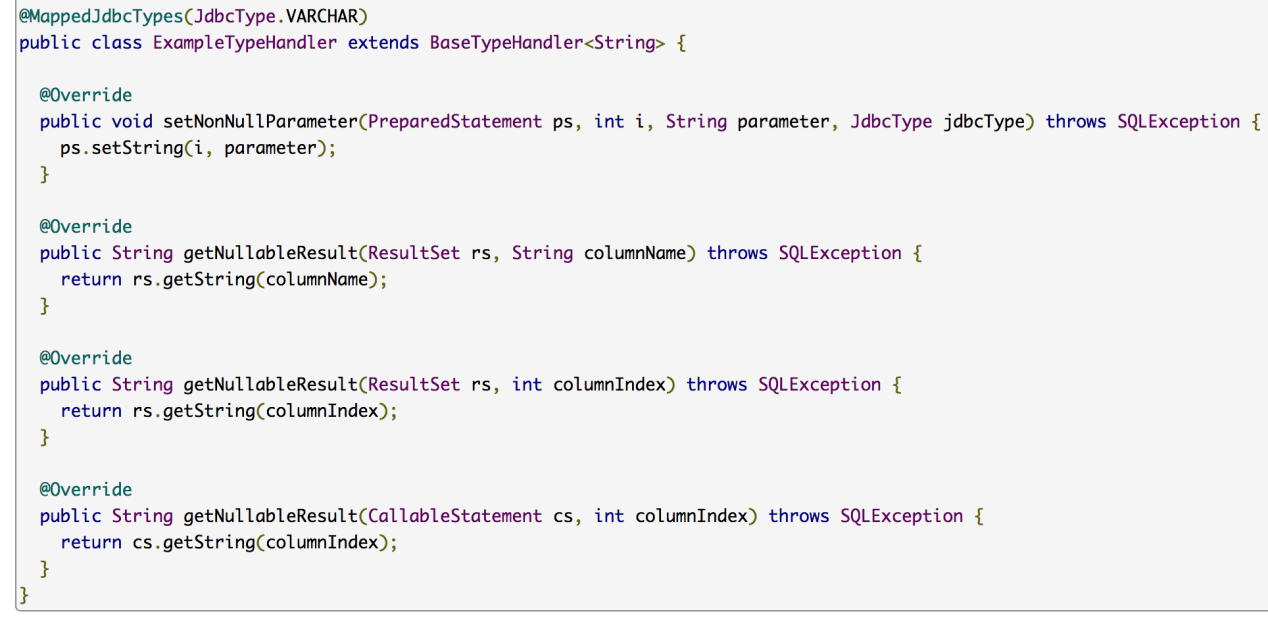
类型别名可以为java类型设置一个缩写名字，不用写全限定名。比如：

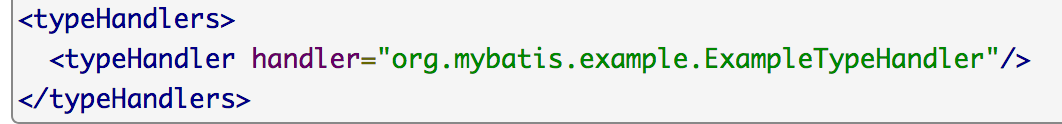


可以指定包名，包下的Javabean会使用首字母小写的类型做为类的别名，如果有注解@Alias，则是注解指定的值。

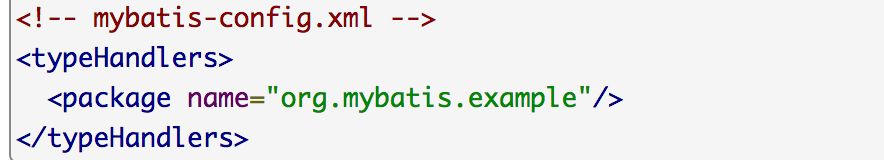


包下的java bean会使用首字母小写的类名作为别名。可以使用@Alias()指定别名；mybatis在进行数据库读写操作时，都需要对java类型与数据库做类型转换，所以其内包含了很多的内置的类型处理器，typeHandlers；有哪些类型处理器可以看参考文档；如果要转化为自定义类型，可以实现org.apache.ibatis.type.TypeHandler接口或者继承org.apache.ibatis.type.BaseTypeHandler基类。比如：

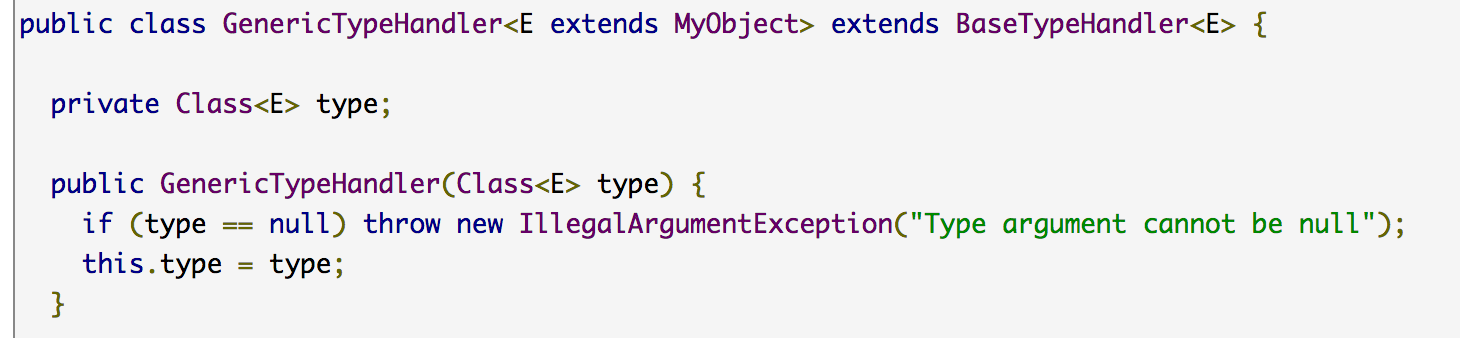




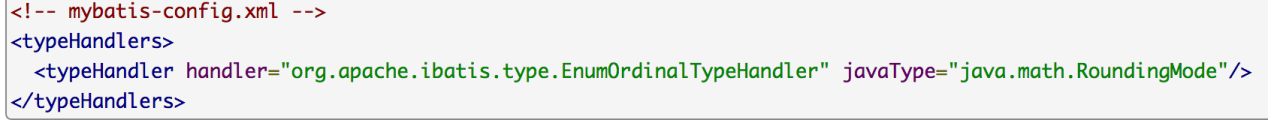
上面范型里面定义的java类型可以通过一定的方式改变，比如在XML元素typehandler 内配置javaType属性，在类型转换类上加入@MappedTypes注解指定类型，同时JDBC类型也可以通过2种方式指定，比如在元素typehandler上通过jdbcType指定或者在类型转换类上指定@MappedJdbcTypes注解，因为Mybatis不会窥探数据库来获得数据类型，所以在ResultMap时，只知道java类型，而不知道jdbc类型，mybatis使用javaType=[类型]，jdbcType=null的标签来选择一个合适的类型转换器，假如加入了@MappedJdbcTypes，注解，那么永远不会被默认的选中，除非显示配置，如果想要默认选中，可以指定includeNullJdbcType=true即可；可以定义一堆类型处理器，指定包就可以（只能通过注解指定JDBC的类型）：



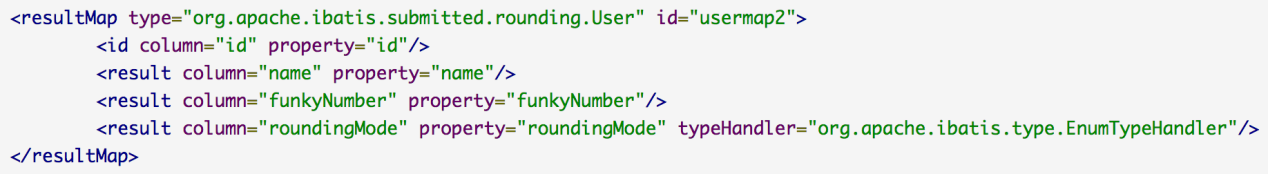
可以定义一个处理多个类型的范型的类型处理器，Mybatis构造时会根据配置的javaType传递一个class



EnumTypeHandler与EnumOrdinalTypeHandler都是范型类型处理器。如果想要映射枚举类型Enum，需要选一个来用，使用EnumTypehandler时保存的事枚举的名字，使用EnumOrdinalTypeHandler时，映射的事序号值。



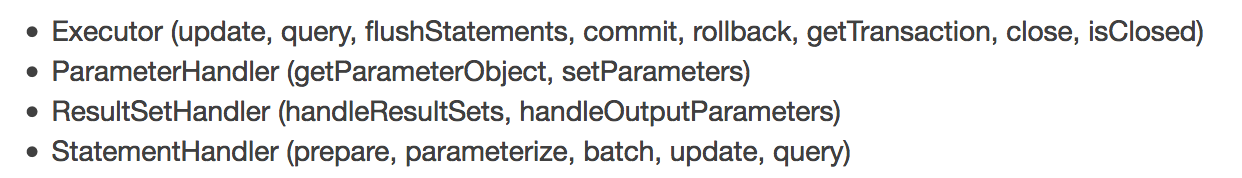
可以在resultMap指定类型转换器，比如：



Mybatis使用对象工厂创建返回结果的POJO对象，也可以创建自己的对象工厂，，比如：



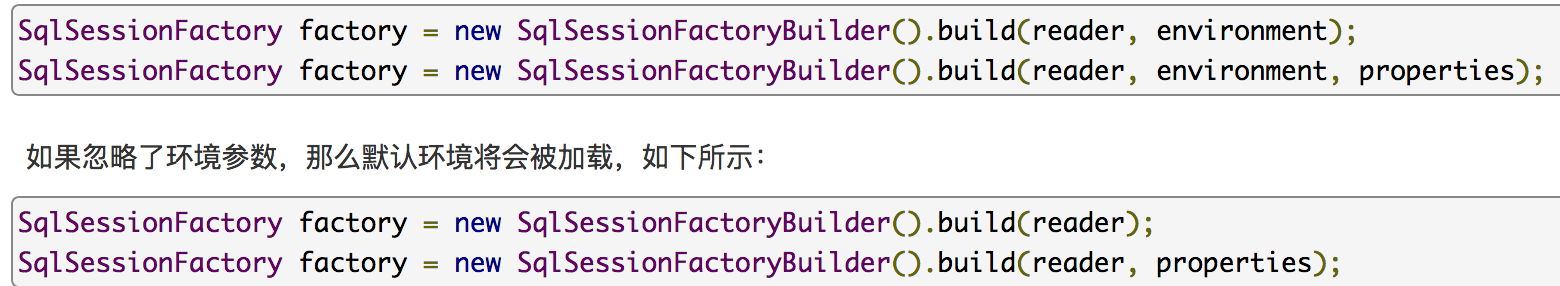
Mybatis允许在语句执行的过程中某一点进行拦截调用，拦截的方式是插件，可以拦截的方法调用有：



插件只需要实现Intercepter接口，并指定想要拦截的方法签名。例子如下：



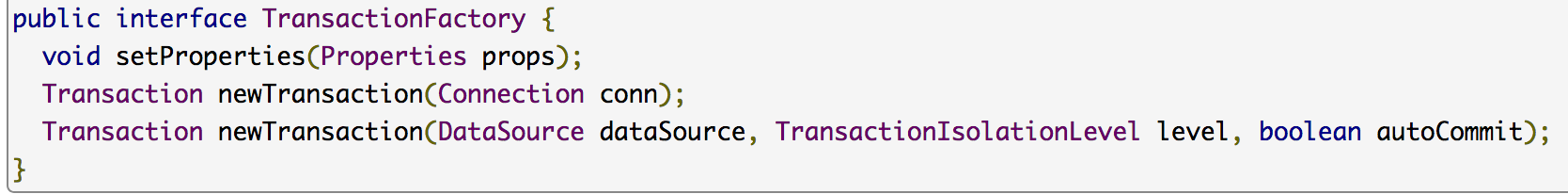
插件可以影响Mybatis的核心行为，覆盖配置类也能达到目的。Mybatis可以配置成适应多种环境，例如开发测试环境的不同，或者多个生产数据库的schema都是相同的，想要复用SQL；但是每个SqlSessionFactory只能选择一个环境（数据库）；指定创建哪种环境，只需要传递environment到SqlSessionBuilder就可以。



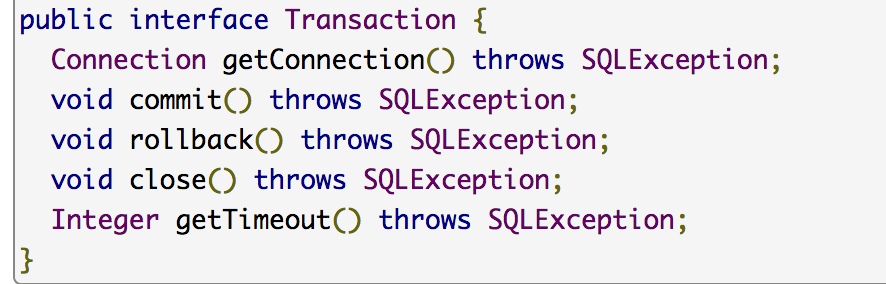
环境元素定义了环境配置：



Mybatis的事务管理器有2种类型，type=[JDBC|MANAGED]，JDBC模式使用了JDBC的提交与回滚，依赖于从数据源得到的连接来管理事务作用域，MANAGED不提交与回滚，让容器来做，默认会关闭连接，需要设置closeConnection=false；如果用的是spring+mybatis，不需要配置事务管理器；可以定义自己的事务管理器，实现TransactionFactory接口就可以，比如：



同时也需要创建一个Transaction接口的实现类，如下：



dataSource元素使用JDBC数据源配置方式来配置连接对象的资源，有3种内建的数据源类型[UNPOOLED|POOLED|JNDI]，UNPOOLED是每次请求都打开或者关闭一次数据库连接，对于简单应用或者数据库连接池不重要的数据库有用，配置的属性额外包含了driver.encoding=UTF8等数据库驱动属性。

POOLED：适用并发，会保存连接对象，避免初始化及认证等：

* PoolMaximumActiveConnections：任意时间最大的连接数量；
* PoolMaximumIdleConnections：任意时间最多的空闲连接数量；
* PoolMaximumCheckoutTime：池中连接被检出使用的最长时间，超过则强制返回；
* PoolTimeToWait：获取链接时间过长打印日志的时间，默认是20秒；
* PoolMaximumLocalBadConnectionTolerance：当获取到一个坏的连接时，允许尝试获取到一个新的连接次数；
* PoolPingQuery：侦测查询，检验连接是否正常；
* PoolPingEnabled：是否启用侦测；
* PoolPingConnectionsNotUsedFor：侦测频率，默认值0，就是一直侦测；

JNDI：外置数据源，这种数据源配置需要2个属性：initial\_context，用来在initialContext中寻找上下文；data\_source；可以添加前缀env.把属性传递给初始化上下文。

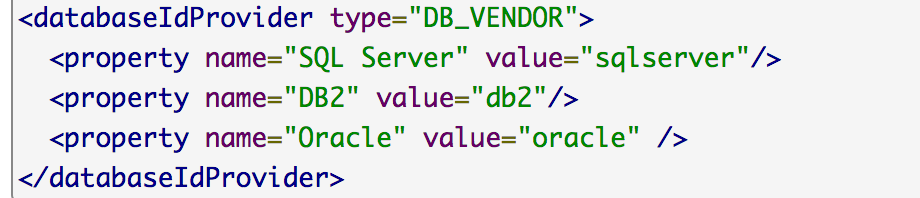




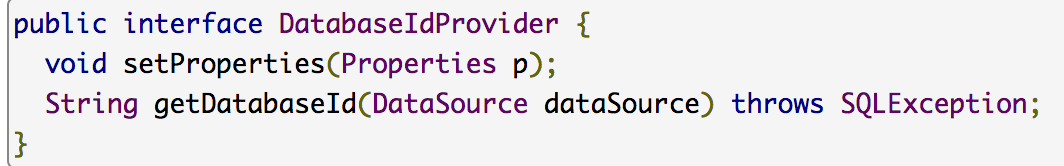
Mybatis可以根据不同的数据库厂商执行不同的语句，辨别不同SQL语句的方式就是databaseId。



设置了DB\_VENDOR属性后，会通过DatabaseMetaData#getDatabase ProductName()返回的字符串进行设置，可以为厂商的名字起别名：



可以实现自己的DatabaseIdProvider，只需要实现一个接口：



最后配置一下映射器所在的目录，可以用classpath目录，file目录等。

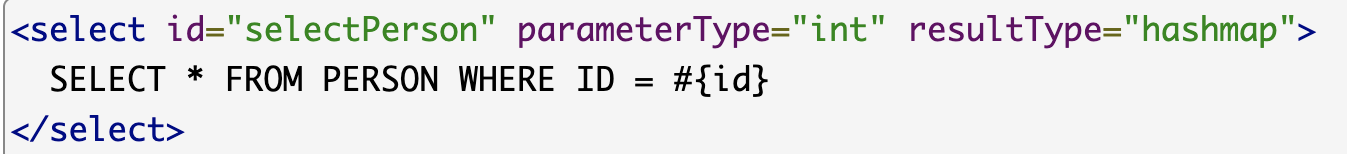
# XML映射器

SQL映射文件只有几个顶级元素：

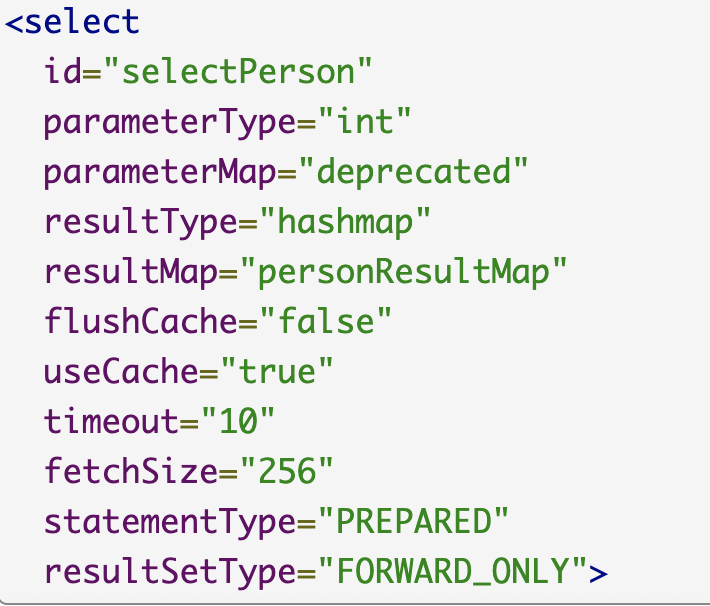
* Cache：该命名空间的缓存配置；
* Cache-ref：引用其他命名空间的缓存配置；
* resultMap：如何从数据库结果集中加载对象，是最复杂的也是最强大的元素；
* sql可以被其他语句引用的可重用语句块；
* Insert：映射插入语句；
* Update：映射更新语句；
* Delete：映射删除语句；
* Select：映射查询语句。

## 4.1 select

最常用，对查询与结果映射做了很多针对JDBC的更好的使用方式。



#{id}会生成一个预编译语句，使用?代替#{}部分，然后set值，配置细节：



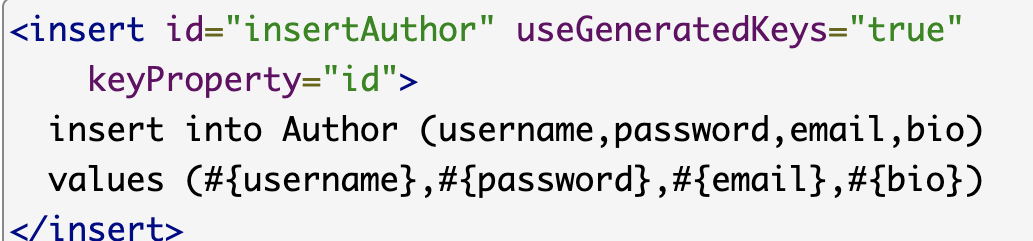
|  |  |
| --- | --- |
| 属性 | 描述 |
| id | 标识符 |
| parameterType | 参数的类全限定名或者别名，这个属性是可选的，MyBatis会通过类型处理器推断出具体传入语句的参数，默认值是unset |
| resultType | 期望从这条语句中返回结果的全限定名或别名，如果是集合，只能是集合包含的类型 |
| resultMap | 对外部resultMap的命名引用 |
| flushCache | 设置为true后，只要语句被调用，都会导致本地缓存与二级缓存被清空 |
| useCache | 设置为true，本条语句的结果被二级缓存缓存起来，select元素的默认值是true |
| timeout | 等待数据库请求结果超时的秒数 |
| fetchSize | 尝试让驱动程序每次批量返回的结果行数 |
| statementType | 可选STATEMENT、PREPARED、CALLABLE，会让MyBatis分别使用Statement、PreparedStatement或CallableStatement |
| resultSetType | FORWARD\_ONLY，SCROLL\_SENSITIVE、SCROLL\_INSENSITIV或者DEAFAULT中的一个 |
| databaseId | 厂商ID |
| resultOrdered |  |
| resultSets | 多结果集 |

## 4.2 insert/update/delete

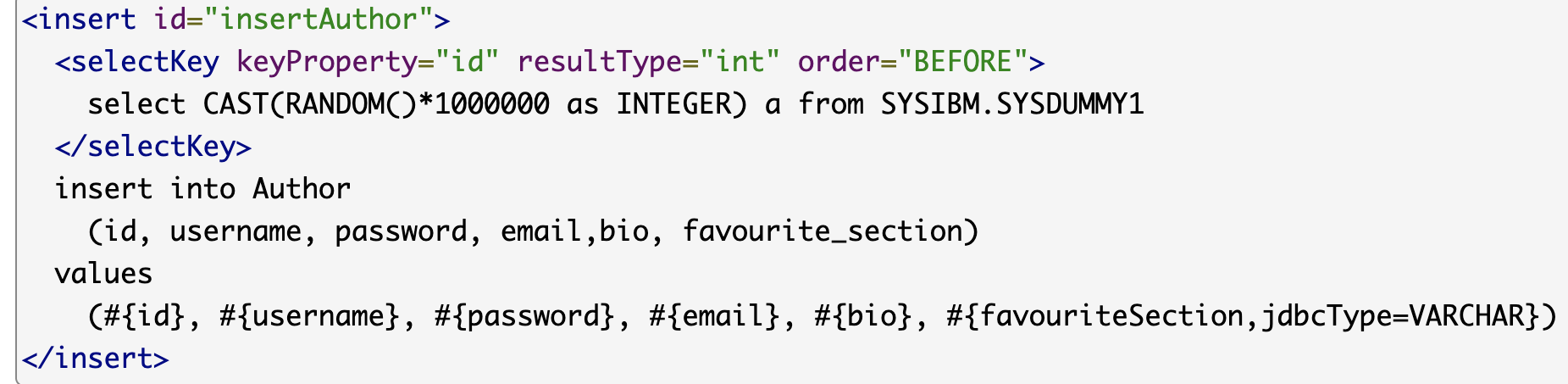
3种语句差不多，含有的比select多的属性如下：

|  |  |
| --- | --- |
| 属性 | 描述 |
| useGeneratedKeys | 仅用于insert/update，这会令Mybatis使用JDBC的getGeneratedKeys取出由数据库内部生成的主键。 |
| keyProperty | 仅用于insert/update，指定识别对象的属性，Mybatis会使用getGeneratedKeys的返回值或insert语句的selectKey子元素设置它的值，可以指定多个属性 |
| keyColumn | 仅用于insert/update，指定生成健值在表中的列名 |

对于可以自动生成主键的数据库来说，可以获得生成的主键

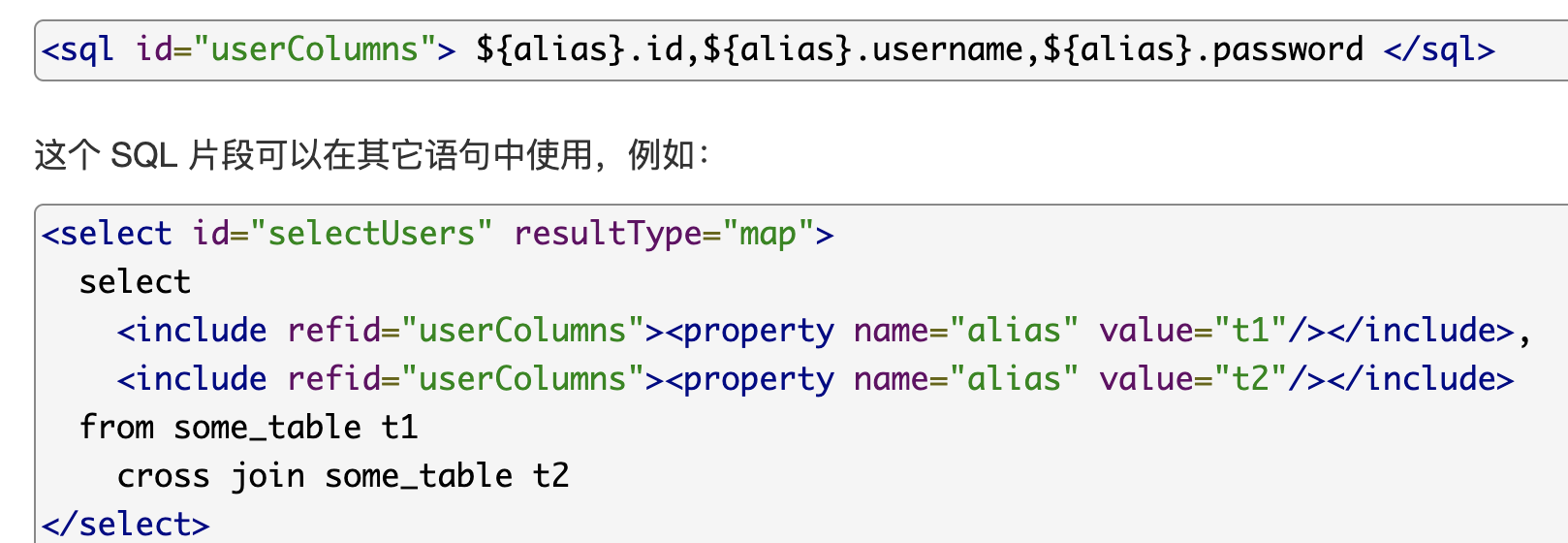


对与不支持自动生成主键的数据库或者JDBC驱动，可以使用selectKey的方式生成key：

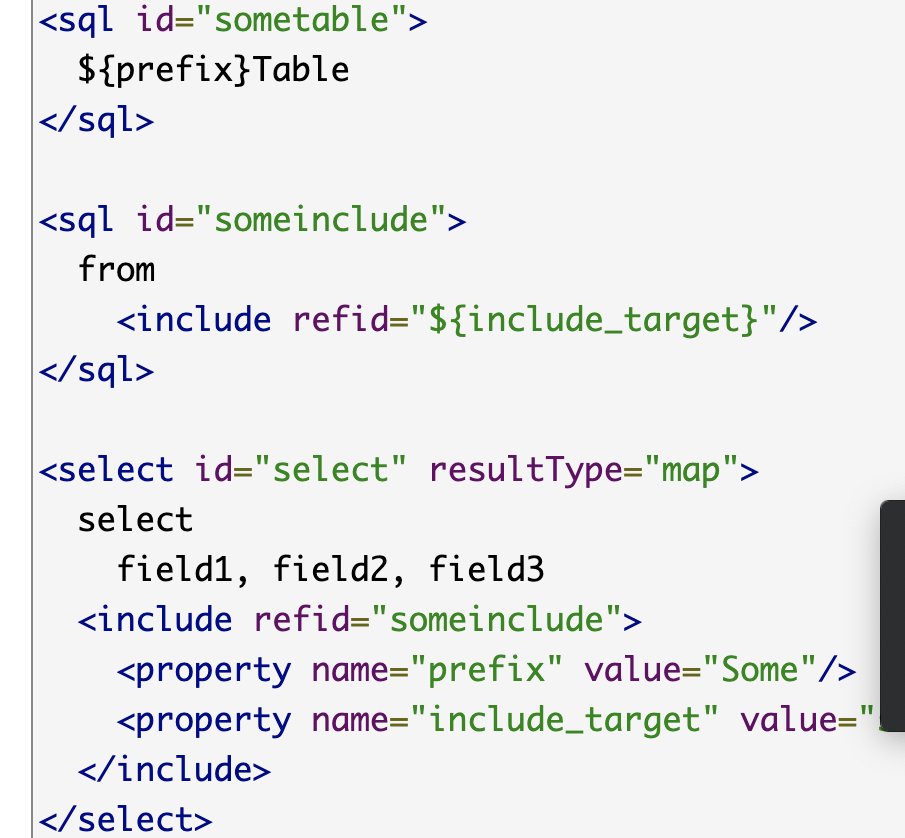


## 4.3 sql

设置可重用的代码段，可以在语句中使用参数，使用include的时候，给出参数，得到实际的sql片段：



refid也可以参数化：



## 4.4 参数

* + 简单的参数形式：传入普通的不可分解的值么，不需要指定parameterType，比如string、int等，可以通过值推断出来；
  + 复杂的对象，比如map：会在对象中寻找；

参与也可以有更复杂的指定方式，比如：

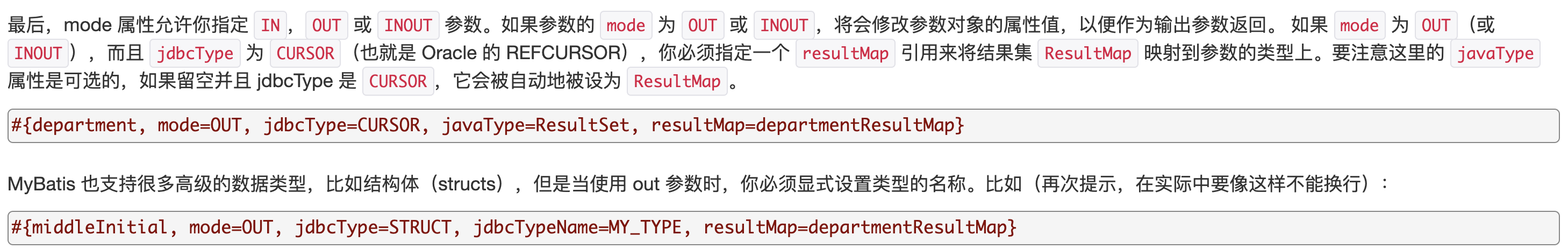


javaType类型Mybatis可以自己推断，除非是HashMap；JDBC要求，如果一个列允许使用null值，并且会使用值为null的参数，必须指定JDBC类型jdbcType，PreparedStatement.setNull()可以获取更多的信息。



对于数值类型，可以指定小数点后面的位数：



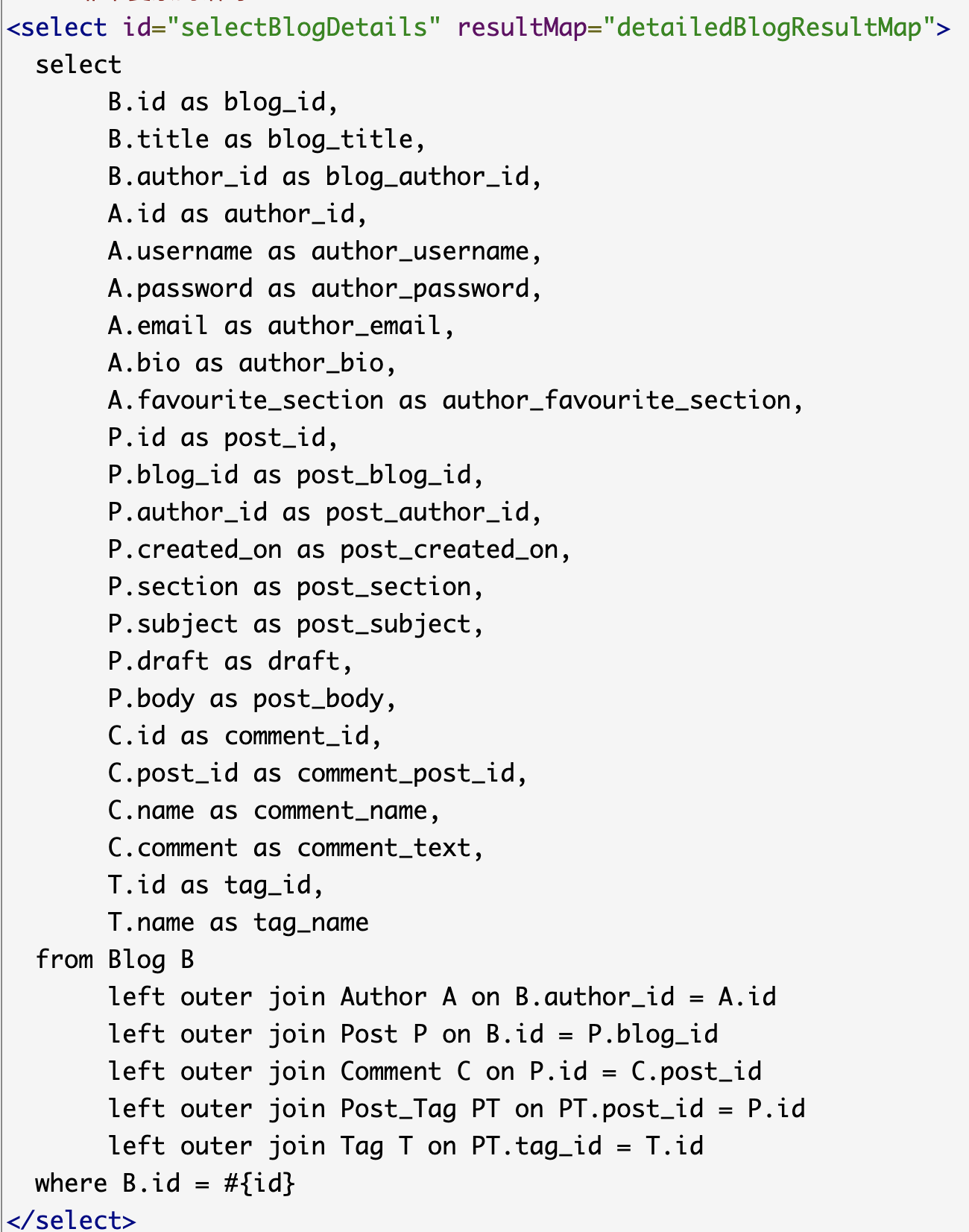


* 使用#{}时，Mybatis会创建PreparedStatement的参数占位符，安全的设置参数，如果要直接插入一个不转义的字符串，比如order by 子句，使用${}符号；如果SQL中的表名或者列名是动态生成的时候，字符串替换非常有用

## 4.5 结果映射

resultMap是Mybatis中最重要最强大的元素，使用ResultType的形式，Mybatis会自动根据Type的类型创建一个ResultMap；主要是用来做名字映射。

结果集映射表示非常复杂的映射关系：





* + Constructor：指定实例化结果集对应的类时使用构造函数的方式生成
    - idArg：使用数据的ID列作为参数；
    - Arg：使用普通的列的值作为参数。
* Id：id列与javaBean属性的对应关系；
* Result：普通列与javaBean属性的对应你关系；
* Association：嵌套结果映射，结果可以是其他的javaType或者resultMap，一般是把某些数据整合成一个对象属性；
* Collection：嵌套结果映射的集合。
* Discriminator：使用结果值来决定使用哪个resultMap；
  + Case：基于值的结果映射。

ResultMap的属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| id | 用于标识映射标识 |
| type | 目标java类型，可以是全限定名或者别名系统定义的别名 |
| autoMapping | 开启/关闭自动映射，会覆盖全局设置 |

Id/Result的属性：

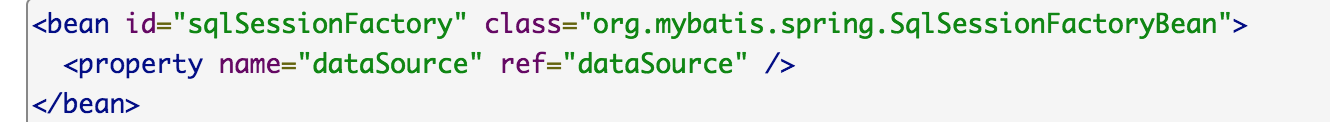
|  |  |
| --- | --- |
| 属性 | 描述 |
| property | java类型中的属性名，支持点号递归查找 |
| column | 数据库中的列名或者SQL中定义的别名 |
| javaType | 目标属性的java类型，可以推断，如果是Hashmap，必须指定不能推断 |
| jdbcType | 只需要在可能需要插入/更新/删除且允许null值时指定 |
| typeHandler | 类型处理器 |

# 第九章 mybatis-spring

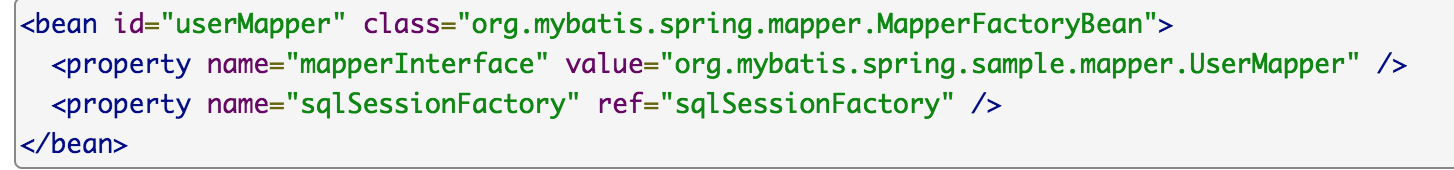
Mybatis-Spring完成了2个框架的整合，Spring将会加载必要的Mybatis工厂类与session类；这个类库也提供了简单的方式来注入Mybatis数据映射器与SqlSession到业务层的bean中。

# 第十章 入门

首先在项目里面加入mybatis-spring-xxxx.jar包；要一起使用的话，需要在spring应用上下文中定义SqlSessionFactory与数据映射器bean；使用SqlSessionFactoryBean构造SqlSessionFactory。



定义数据映射器Bean：



此时可以将这个mapper bean注入到任意的对象中，MapperFactoryBean处理SqlSession的创建与关闭与事务操作，并且任何异常都会被翻译成DataAccess Exception异常。

# 第十一章 SqlSessionFactoryBean

基本Mybatis中，session工厂使用SqlSessionFactoryBuilder构造，在与Spring整合时，使用SqlSessionFactoryBean代替。

SqlSessionFactoryBean实现了Spring的FactoryBean接口，所以定义的bean返回的不是SqlSessionFactoryBean本身而是getObject的结果；在整合中，不需要关注SqlSessionFactory，这个工厂将会被注入到MapperFactoryBean或者扩展了SqlSessionDaoSupport的DAO中。

SqlSessionFactory必须有datasource属性，还有configLocation属性，这个属性指定Mybatis的基础配置的文件路径，这个配置文件不需要配置所有的Mybatis基础配置，整合框架只关注<settings>与<typeAliases>部分，其他的在构造SqlSessionFactoryBean时就确定了。

如果XML映射文件与映射器类不在同包下，需要在XML配置文件中用<mappers>元素指定或者使用SqlSessionFactory的mapperLocations属性。



指定的路径时Ant样式的。

# 第十二章 事务