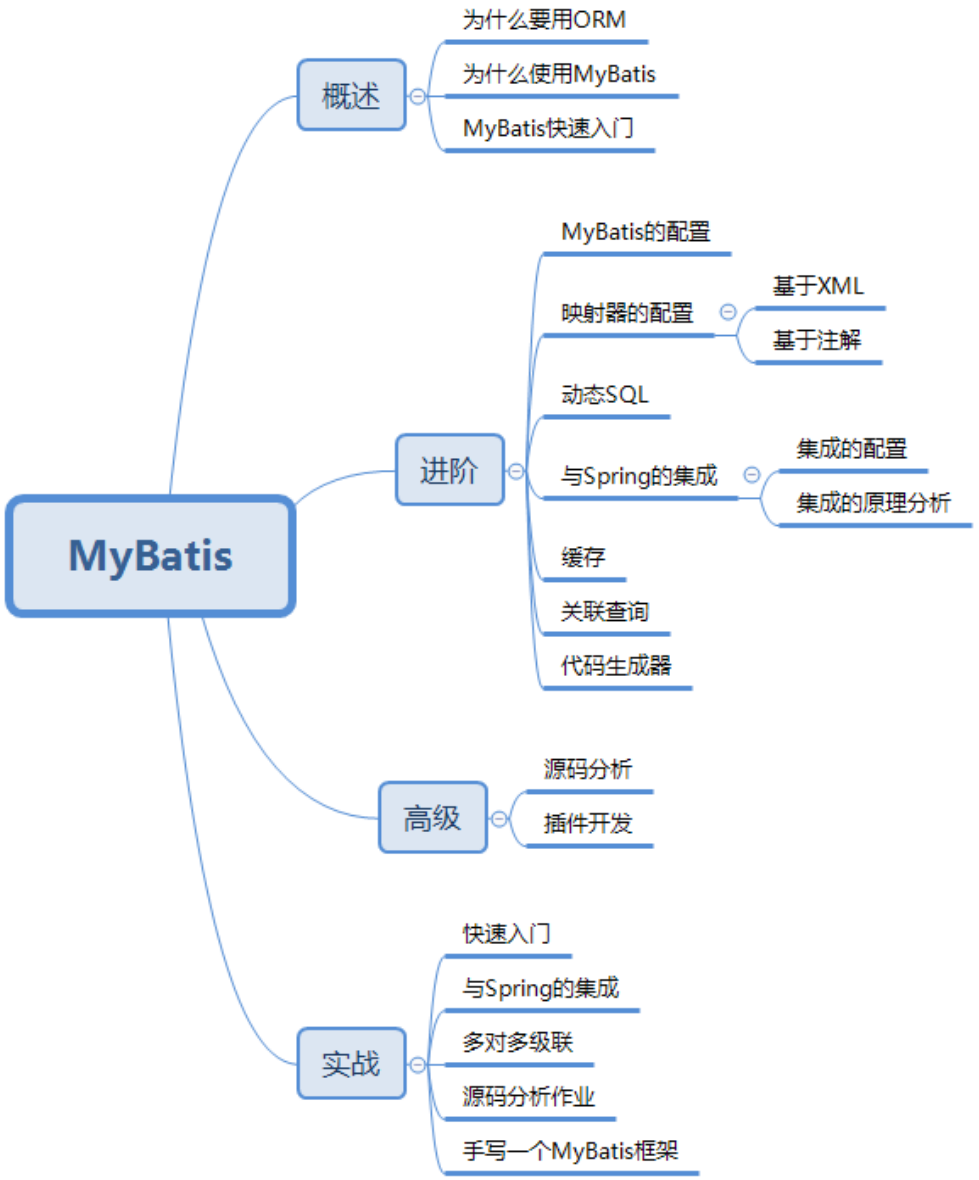
Mybatis概述与进阶

T A H N K Y O U F O R W A T C H I N G

主讲老师Lison：525765982

课程咨询依娜老师：2470523467

概述


MyBatis
概述
为什么要用ORM
为什么使用MyBatis
MyBatis快速入门
进阶
MyBatis的配置
映射器的配置
基于XML
基于注解
动态SQL
与Spring的集成
集成的配置
集成的原理分析
缓存
关联查询
代码生成器
高级
源码分析
插件开发
实战
快速入门
与Spring的集成
多对多级联
源码分析作业
手写一个MyBatis框架

目 录

CONTENTS

概述

为什么要用ORM

mybatis的特点

快速入门

Mybatis配置

mybatis的xml文件详解

mapper的配置

- ✓ 基于xml的配置
- ✓ 基于注解的配置

动态SQL

动态拼装sql

为什么需要ORM框架？

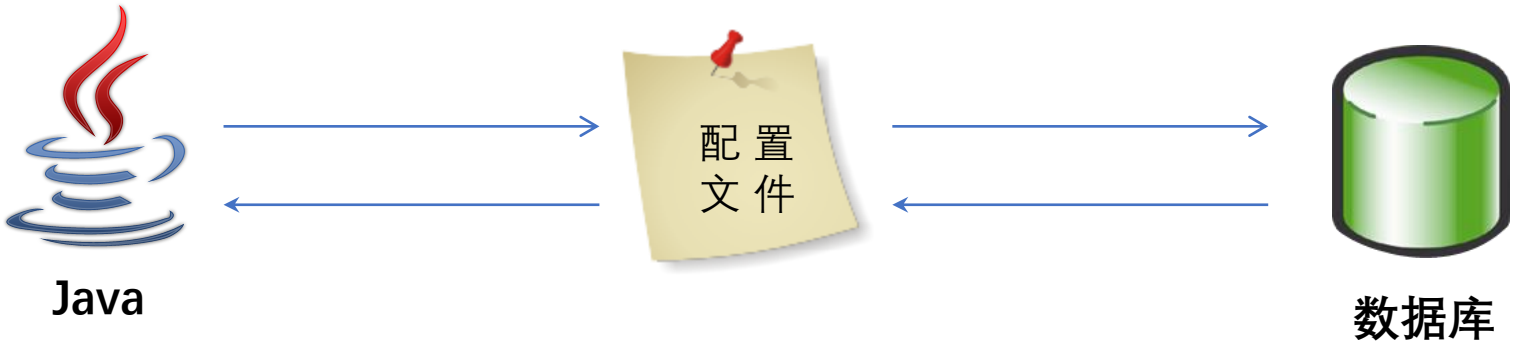
先来看一段JDBC的代码！

传统的JDBC编程存在的弊端：

- ✓ 工作量大，操作数据库至少要5步；
- ✓ 业务代码和技术代码耦合；
- ✓ 连接资源手动关闭，带来了隐患；

ORM是什么？

- **对象关系映射（ORM Obeject Relational Mapping），**ORM模型就是数据库的表与简单Java对象（POJO）的映射模型，它主要解决数据库数据和POJO对象的相互映射；



ORM带来的好处：

- ✓ 更加贴合面向对象的编程语意，Java程序员喜欢的姿势；
- ✓ 技术和业务解耦，Java程序员无需对数据库相关的知识深入了解
- ✓ 妈妈再也不用担心我，不释放数据库连接资源了

ORM框架两大霸主

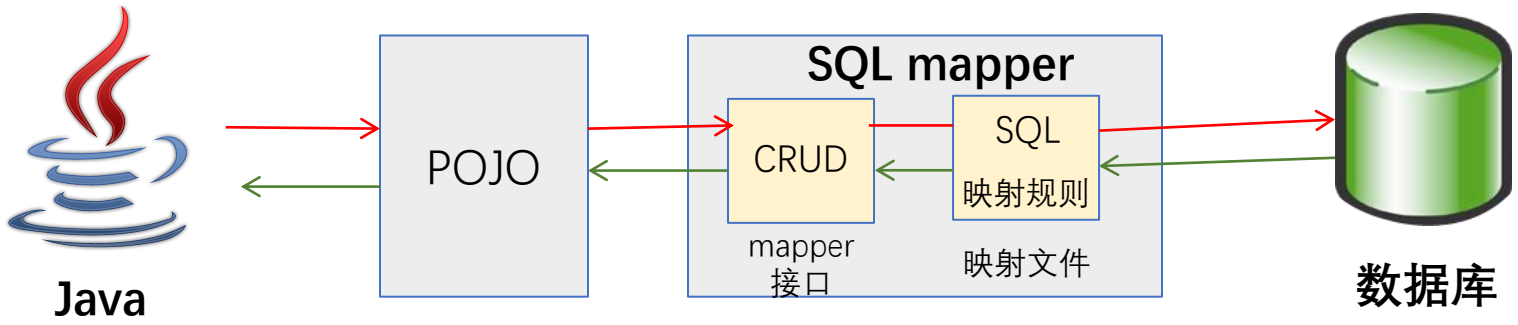
VS

- ✓ 全表映射带来不便
- ✓ 无法自定义组装SQL
- ✓ 复杂关联关系以及复杂SQL语句支持弱
- ✓ 不支持存储过程
- ✓ HQL黑盒封装，调优复杂
- ✓ 性能较差，不适合大型互联网高性能要求

- ✓ 几乎可以替换JDBC
- ✓ 高度灵活
- ✓ 基于底层SQL的优化能力
- ✓ 学习门槛低，易于维护
- ✓ 开发工作量相对较大

Mybatis是什么？

- **Mybatis**前身是iBatis,其源于“Internet”和“ibatis”的组合，本质是一种半自动的ORM框架，除了POJO和映射关系之外，还需要编写SQL语句；



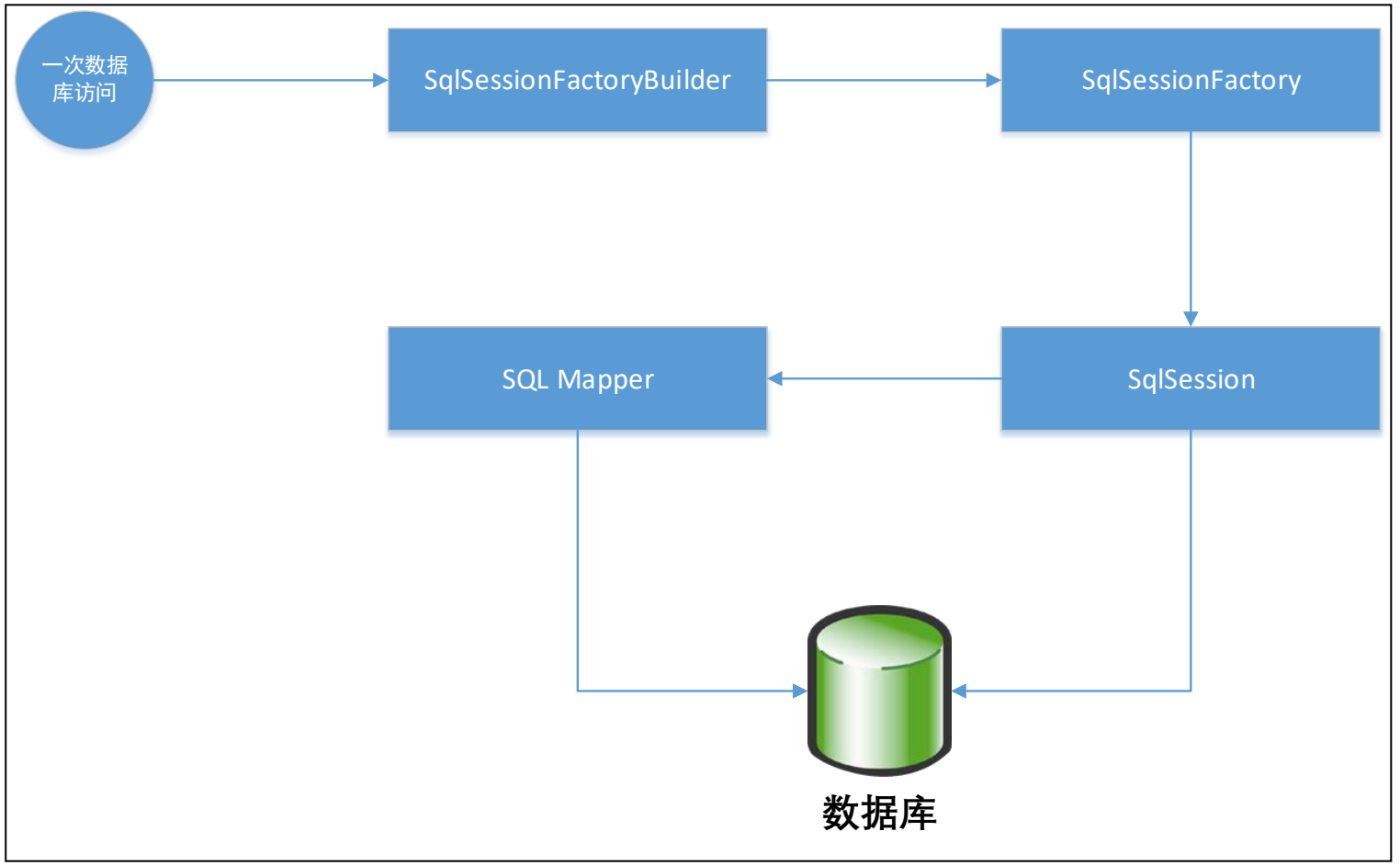
Mybatis映射文件三要素：

- ✓ SQL
- ✓ 映射规则
- ✓ POJO

Mybatis快速入门

1. 加入mybatis的依赖
2. 添加mybatis的配置文件
3. 场景介绍
4. 编写实体类、mapper接口以及mapper xml文件；
5. 编写实例代码

Mybatis快速入门



- **SqlSessionFactoryBuilder：**读取配置信息创建SqlSessionFactory，建造者模式，方法级别生命周期；
- **SqlSessionFactory：**创建Sqlsession，工厂单例模式，存在于程序的整个生命周期；
- **SqlSession：**代表一次数据库连接，可以直接发送SQL执行，也可以通过调用Mapper访问数据库；线程不安全，要保证线程独享（方法级）；
- **SQL Mapper：**由一个Java接口和XML文件组成，包含了要执行的SQL语句和结果集映射规则。方法级别生命周期；

目 录

CONTENTS

概述

为什么要用ORM

mybatis的特点

快速入门

Mybatis配置

mybatis的xml文件详解

mapper的配置

- ✓ 基于xml的配置
- ✓ 基于注解的配置

动态SQL

动态拼装sql

Mybatis配置

序号	属性名	说明	备注
1	properties	定义配置，配置的属性可以在整个配置文件中其他位置进行引用；	重要，优先使用property配置文件解耦
2	settings	设置，用于指定MyBatis的一些全局配置属性，这些属性非常重要，它们会改变MyBatis的运行时行为；	重要，后面专门说明
3	typeAliases	别名，为Java类型设置一个短的名字，映射时方便使用；分为系统定义别名和自定义别名；	可以通过xml和注解配置
4	typeHandlers	用于jdbcType与javaType之间的转换；	无特殊需求不需要调整；后面专题说明
5	ObjectFactory	MyBatis每次创建结果对象的新实例时，它都会使用对象工厂（ObjectFactory）去构建POJO	大部分场景下无需修改
6	plugins	插件，MyBatis允许你在已映射的语句执行过程中的某一点进行拦截调用；	后面专题说明
7	environments	用于配置多个数据源，每个数据源分为数据库源和事务的配置；	在多数据源环境使用
8	databaseIdProvider	MyBatis可以根据不同的数据库厂商执行不同的语句，用于一个系统内多厂商数据源支持。	大部分场景下无需修改
9	mappers	配置引入映射器的方法。可以使用相对于类路径的资源引用、或完全限定资源定位符（包括file:///的URL），或类名和包名等等	后面会专题说明

Mybatis配置 setting（1）

设置参数	描述	有效值	默认值
cacheEnabled	该配置影响的所有映射器中配置的缓存的全局开关	true | false	true
lazyLoadingEnabled	延迟加载的全局开关。当开启时，所有关联对象都会延迟加载。 特定关联关系中可通过设置fetchType属性来覆盖该项的开关状态	true | false	false
aggressiveLazyLoading	当启用时，对任意延迟属性的调用会使带有延迟加载属性的对象完整加载；反之，每种属性将会按需加载。	true | false	true
multipleResultSetsEnabled	是否允许单一语句返回多结果集（需要兼容驱动）。	true | false	true
useColumnLabel	使用列标签代替列名。不同的驱动在这方面会有不同的表现， 具体可参考相关驱动文档或通过测试这两种不同的模式来观察所用驱动的结果。	true | false	true
useGeneratedKeys	允许 JDBC 支持自动生成主键，需要驱动兼容。 如果设置为 true 则这个设置强制使用自动生成主键，尽管一些驱动不能兼容但仍可正常工作（比如 Derby）。	true | false	False
autoMappingBehavior	指定 MyBatis 应如何自动映射列到字段或属性。 NONE 表示取消自动映射；PARTIAL 只会自动映射没有定义嵌套结果集映射的结果集。 FULL 会自动映射任意复杂的结果集（无论是否嵌套）。	NONE, PARTIAL, FULL	PARTIAL

Mybatis配置 setting（2）

设置参数	描述	有效值	默认值
defaultExecutorType	配置默认的执行器。SIMPLE 就是普通的执行器；REUSE 执行器会重用预处理语句（prepared statements）； BATCH 执行器将重用语句并执行批量更新。	SIMPLE、REUSE、BATCH	SIMPLE
defaultStatementTimeout	设置超时时间，它决定驱动等待数据库响应的秒数。	Any positive integer	Not Set (null)
safeRowBoundsEnabled	允许在嵌套语句中使用分页（RowBounds）。	true | false	False
mapUnderscoreToCamelCase	是否开启自动驼峰命名规则（camel case）映射，即从经典数据库列名 A_COLUMN 到经典 Java 属性名 aColumn 的类似映射。	true | false	False
localCacheScope	MyBatis 利用本地缓存机制（Local Cache）防止循环引用（circular references）和加速重复嵌套查询。 默认值为 SESSION，这种情况下会缓存一个会话中执行的所有查询。 若设置值为 STATEMENT，本地会话仅用在语句执行上，对相同 SqlSession 的不同调用将不会共享数据。	SESSION | STATEMENT	SESSION
jdbcTypeForNull	当没有为参数提供特定的 JDBC 类型时，为空值指定 JDBC 类型。某些驱动需要指定列的 JDBC 类型，多数情况直接用一般类型即可，比如 NULL、VARCHAR 或 OTHER。	JdbcType 枚举，最常见的是: NULL, VARCHAR and OTHER	OTHER

Mybatis配置 setting（3）

设置参数	描述	有效值	默认值
lazyLoadTriggerMethods	指定哪个对象的方法触发一次延迟加载。	如果是方法列表用逗号隔开；	equals,clone,hashCode,toString
callSettersOnNulls	指定当结果集中值为 null 的时候是否调用映射对象的 setter（map 对象时为 put）方法，这对于有 Map.keySet() 依赖或 null 值初始化的时候是有用的。注意基本类型（int、boolean 等）是不能设置成 null 的。	true | false	false
logPrefix	指定 MyBatis 增加到日志名称的前缀。	Any String	Not set
logImpl	指定 MyBatis 所用日志的具体实现，未指定时将自动查找。	SLF4J | LOG4J | LOG4J2 | JDK_LOGGING | COMMONS_LOGGING | STDOUT_LOGGING | NO_LOGGING	Not set
proxyFactory	指定 Mybatis 创建具有延迟加载能力的对象所用到的代理工具。	CGLIB | JAVASSIST	版本3.3.0以上JAVASSIST

Mybatis配置 environments

- **environment** 元素是配置一个数据源的开始，属性id是它的唯一标识

- **transactionManager** 元素配置数据库事务，其中type属性有三种配置方式
  - ✓ jdbc，采用jdbc的方式管理事务；
  - ✓ managed，采用容器的方式管理事务，在JNDI数据源中使用；
  - ✓ 自定义，自定义数据库事务管理办法；

- **dataSource** 元素配置数据源连接信息，type属性是连接数据库的方式配置，有四种配置方式
  - ✓ UNPOOLED 非连接池方式连接
  - ✓ POOLED　使用连接池连接
  - ✓ JNDI　使用JNDI数据源
  - ✓ 自定义数据源

Mybatis配置 mapper

PS：第一种方式用的推荐使用，类文件和mapper文件可以不需要放在一个文件夹中，xml文件也不会和java文件混合在一起；

- **用classPath下资源引用**

```
<mappers>
    <!--直接映射到相应的mapper文件 -->
    <mapper resource="sqlmapper/TUserMapper.xml" />
</mappers>
```

- **用类注册方式引用**

```
<mappers>
    <!–通过类扫描mapper文件 -->
    <mapper class="com.enjoylearning.mybatis.mapper.TUserMapper" />
</mappers>
```

- **使用包名引入引射器名**

```
<mappers>
    <!–扫描包下所有的mapper文件 -->
    <package name="com.enjoylearning.mybatis.mapper"/>
</mappers>
```

- **用文件的全路径引用**

目 录

CONTENTS

概述

为什么要用ORM

mybatis的特点

快速入门

Mybatis配置

mybatis的xml文件详解

mapper的配置

- ✓ 基于xml的配置
- ✓ 基于注解的配置

动态SQL

动态拼装sql

基于xml配置的映射器

- cache – 给定命名空间的缓存配置。
- cache-ref – 其他命名空间缓存配置的引用。
- resultMap – 是最复杂也是最强大的元素，用来描述如何从数据库结果集中来加载对象。
- sql – 可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

select元素

元素	说明	备注
id	它和Mapper的命名空间组合起来是唯一的，提供给MyBatis调用	如果命名空间和id组合起来不唯一，会抛出异常
parameterType	传入参数的类型；可以给出类全名，也可以给出类别名，使用别名必须是MyBatis内部定义或自定义的； 基本数据类型：int，String，long，date(不知是sql.date 还是 util.date) 复杂数据类型：类 和 Map	可以选择JavaBean，Map等复杂的参数类型传递给SQL
resultType	从这条语句中返回的期望类型的类的完全限定名或别名。 注意如果是集合情形，那应该是集合可以包含的类型，而不能是集合本身。 使用 resultType 或 resultMap，但不能同时使用 定义类的全路径，在允许自动匹配的情况下，结果集将通过JavaBean的规范映射； 或者定义为int,double,float等参数... 也可以使用别名，但是要符合别名规范，不能和resultMap同时使用。	它是我们常用的参数之一，比如我们总计总条数 就可以把它的值设为int
resultMap	外部 resultMap 的命名引用。使用 resultMap 或 resultType，但不能同时使用； 它是映射集的引用，将执行强大的映射功能，我们可以使用resultType或者resultMap其中的一个，resultMap可以给予我们自定义映射规则的机会	它是MyBatis最复杂的元素，可以配置映射规则，级联，typeHandler等
flushCache	它的作用是在调用SQL后，是否要求MyBatis清空之前查询的本地缓存和二级缓存	true/false，默认为false
useCache	启动二级缓存开关，是否要求MyBatis将此次结果缓存	true/false，默认为true
timeout	设置超时时间，超时之后抛出异常，秒	默认值为数据库厂商提供的JDBC驱动所设置的秒数
fetchSize	获取记录的总条数设定	默认值是数据库厂商提供的JDBC驱动所设的条数

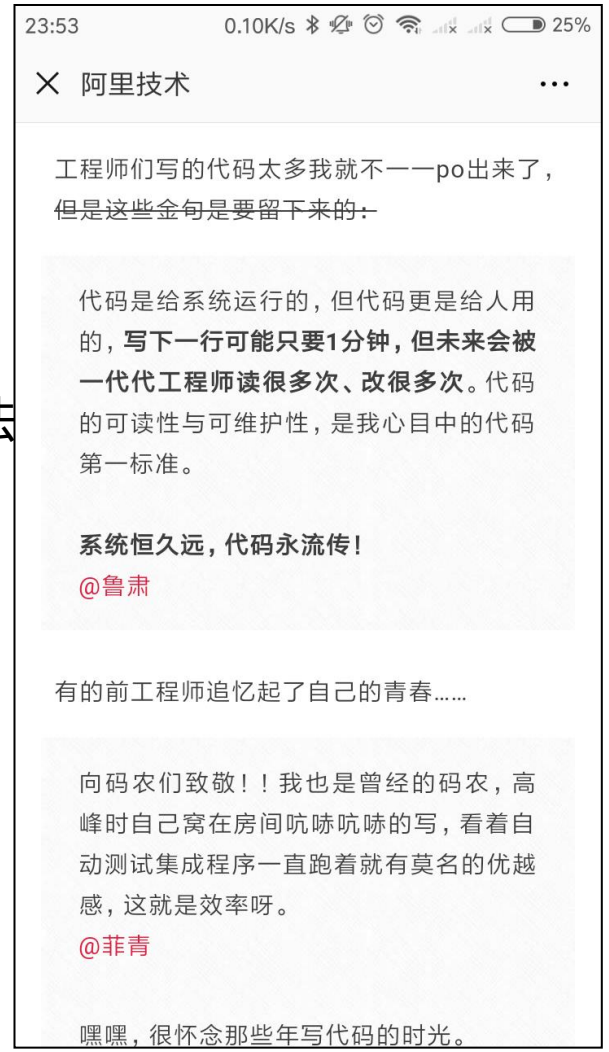
select元素

自动映射

- ✓ 前提：SQL列名和JavaBean的属性是一致的；
- ✓ 自动映射等级autoMappingBehavior设置为PARTIAL，需要谨慎使用FULL；
- ✓ 使用resultType；
- ✓ 如果列名和JavaBean不一致，但列名符合单词下划线分割，Java是驼峰命名法则mapUnderscoreToCamelCase可设置为true；

传递多个查询入参

- ✓ 使用map传递参数；可读性差，导致可维护性和可扩展性差，杜绝使用；
- ✓ 使用注解传递参数；直观明了，当参数较少一般小于5个的时候，建议使用；
- ✓ 使用Java Bean的方式传递参数；当参数大于5个的时候，建议使用；



resultMap元素 属性

- ✓ resultMap 元素是 MyBatis 中最重要最强大的元素。它可以让你从 90% 的 JDBC ResultSets 数据提取代码中解放出来,在对复杂语句进行联合映射的时候，它很可能可以代替数千行的同等功能的代码。
- ✓ ResultMap 的设计思想是，简单的语句不需要明确的结果映射，而复杂一点的语句只需要描述它们的关系就行了。

属性	描述
id	当前命名空间中的一个唯一标识，用于标识一个result map.
type	类的完全限定名, 或者一个类型别名 (内置的别名可以参考上面的表格).
autoMapping	如果设置这个属性，MyBatis将会为这个ResultMap开启或者关闭自动映射。这个属性会覆盖全局的属性 autoMappingBehavior。默认值为：unset。

resultMap元素 子元素

- constructor - 用于在实例化类时，注入结果到构造方法中
  - idArg - ID 参数;标记出作为 ID 的结果可以帮助提高整体性能
  - arg - 将被注入到构造方法的一个普通结果
- id – 一个 ID 结果;标记出作为 ID 的结果可以帮助提高整体性能
- result – 注入到字段或 JavaBean 属性的普通结果
- association – 一个复杂类型的关联;许多结果将包装成这种类型
  - 嵌套结果映射 – 关联可以指定为一个 resultMap 元素，或者引用一个
- collection – 一个复杂类型的集合
  - 嵌套结果映射 – 集合可以指定为一个 resultMap 元素，或者引用一个
- discriminator – 使用结果值来决定使用哪个 resultMap
  - case – 基于某些值的结果映射
    - 嵌套结果映射 – 一个 case 也是一个映射它本身的结果,因此可以包含很多相 同的元素，或者它可以参照一个外部的 resultMap

id & result

- ✓ id 和 result 都将一个列的值映射到一个简单数据类型(字符串,整型,双精度浮点数,日期等)的属性或字段
- ✓ 两者之间的唯一不同是， id 表示的结果将是对象的标识属性，这会在比较对象实例时用到。 这样可以提高整体的性能，尤其是缓存和嵌套结果映射(也就是联合映射)的时候

属性	描述
property	POJO中映射到列结果的字段或者属性。如果POJO的属性匹配的是存在的，和给定SQL列名（column元素）相同的，那么MyBatis就会自动映射；
column	SQL中的列名,或者是列的别名。一般情况下，这和 传递给 resultSet.getString(columnName) 方法的参数一样。
javaType	配置的Java的类；
jdbcType	配置的数据库的类型；
typeHandler	类型处理器，使用这个属性,你可以覆盖默 认的类型处理器。这个属性值是一个类型处理 器实现类的完全限定名，或者是类型别名。

constructor

- ✓ 一个pojo不存在没有参数的构造方法，就需要使用constructor;
- ✓ 为了通过名称来引用构造方法参数，你可以添加 @Param 注解,指定参数名称的前提下，以任意顺序编写 arg 元素

```
<constructor>
        <idArg column="id" javaType="int" />
        <arg column="user_name" javaType="String" />
</constructor>
```

insert, update 和 delete

属性	描述
id	命名空间中的唯一标识符，可被用来代表这条语句。
parameterType	将要传入语句的参数的完全限定类名或别名。这个属性是可选的，因为 MyBatis 可以通过 TypeHandler 推断出具体传入语句的参数，默认值为 unset。
flushCache	将其设置为 true，任何时候只要语句被调用，都会导致本地缓存和二级缓存都会被清空，默认值：true（对应插入、更新和删除语句）。
timeout	这个设置是在抛出异常之前，驱动程序等待数据库返回请求结果的秒数。默认值为 unset（依赖驱动）。
statementType	STATEMENT，PREPARED 或 CALLABLE 的一个。这会让 MyBatis 分别使用 Statement，PreparedStatement 或 CallableStatement，默认值：PREPARED。
useGeneratedKeys	（仅对 insert 和 update 有用）这会令 MyBatis 使用 JDBC 的 getGeneratedKeys 方法来取出由数据库内部生成的主键（比如：像 MySQL 和 SQL Server 这样的关系数据库管理系统的自动递增字段），默认值：false。
keyProperty	（仅对 insert 和 update 有用）唯一标记一个属性，MyBatis 会通过 getGeneratedKeys 的返回值或者通过 insert 语句的 selectKey 子元素设置它的键值，默认：unset。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。
keyColumn	（仅对 insert 和 update 有用）通过生成的键值设置表中的列名，这个设置仅在某些数据库（像 PostgreSQL）是必须的，当主键列不是表中的第一列的时候需要设置。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。
databaseId	如果配置了 databaseIdProvider，MyBatis 会加载所有的不带 databaseId 或匹配当前 databaseId 的语句；如果带或者不带的语句都有，则不带的会被忽略。

selectKey元素

属性	描述
keyProperty	selectKey 语句结果应该被设置的目标属性。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。
keyColumn	匹配属性的返回结果集中的列名称。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。
resultType	结果的类型。MyBatis 通常可以推算出来，但是为了更加确定写上也不会有什么问题。MyBatis 允许任何简单类型用作主键的类型，包括字符串。如果希望作用于多个生成的列，则可以使用一个包含期望属性的 Object 或一个 Map。
order	这可以被设置为 BEFORE 或 AFTER。如果设置为 BEFORE，那么它会首先选择主键，设置 keyProperty 然后执行插入语句。如果设置为 AFTER，那么先执行插入语句，然后是 selectKey 元素 - 这和像 Oracle 的数据库相似，在插入语句内部可能有嵌入索引调用。
statementType	与前面相同，MyBatis 支持 STATEMENT，PREPARED 和 CALLABLE 语句的映射类型，分别代表 PreparedStatement 和 CallableStatement 类型。

```
<selectKey  keyProperty=“id” order= " Before" resultType="int">
        select SEQ_ID.nextval from dual
</selectKey>
```

sql元素和参数

- **sql元素：**用来定义可重用的 SQL 代码段，可以包含在其他语句中；

- **参数：向sql语句中传递的可变参数**
  - ✓ 预编译 #{}：将传入的数据都当成一个字符串，会对自动传入的数据加一个双引号，能够很大程度防止sql注入；
  - ✓ 传值　${}：传入的数据直接显示生成在sql中，无法防止sql注入；
  - ✓ 表名、选取的列是动态的，order by和in操作， 可以考虑使用$

注解方式配置

- 注解方式就是将SQL语句直接写在接口上，对于需求比较简单的系统，效率较高。缺点在于，每次修改sql语句都要编译代码，对于复杂的sql语句可编辑性和可读性都差，一般不建议使用这种配置方式；
  - ✓ @Select
  - ✓ @Results
  - ✓ @Insert
  - ✓ @Update
  - ✓ @Delete

目 录

CONTENTS

概述

为什么要用ORM

mybatis的特点

快速入门

Mybatis配置

mybatis的xml文件详解

mapper的配置

- ✓ 基于xml的配置
- ✓ 基于注解的配置

动态SQL

动态拼装sql

动态sql元素

元素	作用	备注
if	判断语句	单条件分支判断
choose、when、otherwise	相当于java的case when	多条件分支判断
Trim、where、set	辅助元素	用于处理sql拼装问题
foreach	循环语句	在in语句等列举条件常用，常用于实现批量操作

批量操作

- ✓ 通过foreach动态拼装SQL语句
- ✓ 使用BATCH类型的excutor