# MyBatis快速入门

## 1.1为什么需要ORM框架？

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；

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、映射规则和POJO；

## 1.2 MyBatis快速入门

步骤如下：

1. 加入mybatis的依赖，版本3.5.x
2. 添加mybatis的配置文件，包括MyBatis核心文件和mapper.xml文件
3. 场景介绍：基于t\_user表单数据查询、多数据查询；
4. 编写实体类、mapper接口以及mapper xml文件；
5. 编写实例代码：com.enjoylearning.mybatis.MybatisDemo. quickStart

核心类分析：

1. SqlSessionFactoryBuilder：读取配置信息创建SqlSessionFactory，建造者模式，方法级别生命周期；
2. SqlSessionFactory：创建Sqlsession，工厂单例模式，存在于程序的整个生命周期；
3. SqlSession：代表一次数据库连接，一般通过调用Mapper访问数据库，也可以直接发送SQL执行， ；线程不安全，要保证线程独享（方法级）；
4. 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；

# resultType还是resultMap?

## 2.1 resultType

**resultType**：当使用resultType做SQL语句返回结果类型处理时，对于SQL语句查询出的字段在相应的pojo中必须有和它相同的字段对应，而resultType中的内容就是pojo在本项目中的位置。

自动映射注意事项 :

1. 前提：SQL列名和JavaBean的属性是一致的；
2. 使用resultType，如用简写需要配置typeAliases （别名）；
3. 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；

演示代码：com.enjoylearning.mybatis.MybatisDemo. testAutoMapping

## 2.2 resultMap

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. |
| 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。 |

使用场景总结：1. 字段有自定义的转化规则；2. 复杂的多表查询

演示代码：com.enjoylearning.mybatis.MybatisDemo. testResultMap

## 2.3到底应该用resultType还是resultMap?

强制使用resultMap, 不要用 resultClass 当返回参数，即使所有类属性名与数据库字段一一对应，也需要定义；见《Java开发手册1.5》之5.4.3；

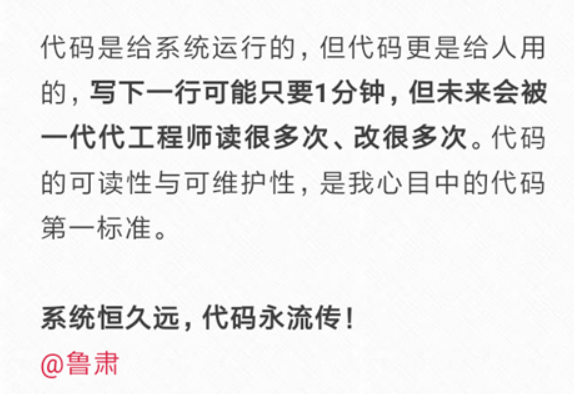
# 怎么传递多个参数？

传递参数有三种方式：

|  |  |
| --- | --- |
| **方式** | **描述** |
| 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 |

建议不要用Map作为mapper的输入和输出，不利于代码的可读性和可维护性；见《Java开发手册1.5》之5.4.6；

演示代码：com.enjoylearning.mybatis.MybatisDemo. testManyParamQuery



# 怎么样获取主键？

## 4.1 通过insert/update标签相关属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| 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。 |

注意：自增长序号不是简单的行数+1，而是序号最大值+1；

示例代码：com.enjoylearning.mybatis.MybatisDemo. testInsertGenerateId1

## 4.2 通过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。 |
| 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，然后获取主键字段；mysql数据库自增长的方式order设置为After，oracle数据库通过sequnce获取主键order设置为Before |

Oracle通过sequnce获取主键示例：

<selectKey keyProperty=*“id” order= " Before" resultType="int">*

select SEQ\_ID.nextval from dual

</selectKey>

Mysql通过自增长序号获取主键示例：

<selectKey keyProperty=*"id"* order=*"AFTER"* resultType=*"int"*>

select LAST\_INSERT\_ID()

</selectKey>

示例代码：com.enjoylearning.mybatis.MybatisDemo. testInsertGenerateId2

# SQL元素和SQL的参数

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；

SQL参数：向sql语句中传递的可变参数,分为预编译#{}和传值${}两种

* 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单引号，能够很大程度防止sql注入；
* 传值${}：传入的数据直接显示生成在sql中，无法防止sql注入；适用场景：动态报表，表名、选取的列是动态的，order by和in操作， 可以考虑使用$

示例代码：com.enjoylearning.mybatis.MybatisDemo.testSymbol

建议：sql.xml 配置参数使用：#{}，#param# 不要使用${} 此种方式容易出现 SQL 注入。见《Java开发手册1.5》之5.4.4；

# 动态SQL

## 6.1 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 |

## 6.2 示例代码说明

|  |  |
| --- | --- |
| **示例代码** | **说明** |
| com.enjoylearning.mybatis.MybatisDemo.testSelectIfOper | 在select中使用if元素，where元素可以在查询条件之前加where关键字，同时去掉语句的第一个and或or |
| com.enjoylearning.mybatis.MybatisDemo.testUpdateIfOper | 在update中使用if元素，set元素可以在值设置之前加set关键字，同时去掉语句最有一个逗号 |
| com.enjoylearning.mybatis.MybatisDemo.testInsertIfOper | 在insert中使用if元素，trim元素可以帮助拼装columns和values |
| com.enjoylearning.mybatis.MybatisDemo.testForeach4In | 使用foreach拼装in条件 |

## 通过Mybatis怎么样进行批量的操作

* 1. 通过foreach动态拼装SQL语句，参考代码见：com.enjoylearning.mybatis.MybatisDemo.testForeach4In. testForeach4Insert
  2. 使用BATCH类型的excutor,参考代码块见：
* com.enjoylearning.mybatis.JdbcDemo. updateDemo，jdbc批处理的原理；
* com.enjoylearning.mybatis.MybatisDemo.testBatchExcutor，基于Mybatis怎么使用Batch类型的excutor；

# 代码生成器

MyBatis Generator：MyBatis 的开发团队提供了一个很强大的代码生成器，代码包含了数据库表对应的实体类 、Mapper 接口类、 Mapper XML 文件等，这些代码文件中几乎包含了全部的单表操作方法，使用 MBG 可以极大程度上方便我们使用 MyBatis，还可以减少很多重复操作；MyBatis Generator的核心就是配置文件，完整的配置文件见：

运行MGB的方式有三种，见下表：

|  |  |  |
| --- | --- | --- |
| **方式** | **运行代码** | **推荐使用场景** |
| 作为 Maven Plugin运行 | mvn mybatis-generator:generate | 对逆向工程定制较多，项目工程结构比较单一的情况 |
| 运行Java 程序使用 XML配置文件 | com.enjoylearning.mybatis.MybatisDemo.mybatisGeneratorTest |
| 从命令提示符 使用 XML 配置文件 | java -jar mybatis-generator-core-x.x.x.jar -configfile generatorConfig.xml  具体见网盘：逆向工程 | 对逆向工程定制较少，项目工程结构比较复杂的情况 |

# 关联查询

## 8.1 关联查询几个需要注意的细节

1. 超过三个表禁止 join。需要 join 的字段，数据类型必须绝对一致；多表关联查询时，保证被关联的字段需要有索引；见《Java开发手册1.5》之5.2.2；
2. 不得使用外键与级联，一切外键概念必须在应用层解决；见《Java开发手册1.5》之5.3.6；
3. 字段允许适当冗余，以提高查询性能，但必须考虑数据一致；见《Java开发手册1.5》之5.1.13；

关联元素：association用于表示一对一关系，collection用于表示一对多关系；

关联方式：

* 嵌套结果:使用嵌套结果映射来处理重复的联合结果的子集
* 嵌套查询:通过执行另外一个 SQL 映射语句来返回预期的复杂类型

## 8.2 一对一关联嵌套结果方式

association标签 嵌套结果方式 常用属性：

* property ：对应实体类中的属性名，必填项。
* javaType ：属性对应的 Java 类型 。
* resultMap ：可以直接使用现有的 resultMap ，而不需要在这里配置映射关系。
* columnPrefix ：查询列的前缀，配置前缀后，在子标签配置 result 的 column 时可以省略前缀

示例代码：com.enjoylearning.mybatis.testOneToOne.

开发小技巧：

1. resultMap可以通过使用extends实现继承关系，简化很多配置工作量；
2. 关联的表查询的类添加前缀是编程的好习惯；
3. 通过添加完整的命名空间，可以引用其他xml文件的resultMap；

## 8.3 一对一关联嵌套查询方式

association标签 嵌套查询方式 常用属性：

* select ：另 一个映射查询的 id, MyBatis 会额外执行这个查询获取嵌套对象的结果 。
* column ：列名（或别名），将主查询中列的结果作为嵌套查询的参数。
* fetchType ：数据加载方式，可选值为 lazy 和 eager，分别为延迟加载和积极加载 ，这个配置会覆盖全局的 lazyLoadingEnabled 配置；

示例代码：com.enjoylearning.mybatis.testOneToOne().

嵌套查询会导致“N+1 查询问题”，导致该问题产生的原因:

1. 你执行了一个单独的 SQL 语句来获取结果列表(就是“+1”)。
2. 对返回的每条记录,你执行了一个查询语句来为每个加载细节(就是“N”)。

这个问题会导致成百上千的 SQL 语句被执行。这通常不是期望的。

解决“N+1 查询问题”的办法就是开启懒加载、按需加载数据，开启懒加载配置：

在<select>节点上配置“fetchType=lazy”

在MyBatis核心配置文件中加入如下配置：

<!-- 开启懒加载 ，当启用时，有延迟加载属性的对象在被调用时将会完全加载任意属性。否则，每种属性将会按需要加载。默认：true -->

<setting name=*"aggressiveLazyLoading"* value=*"false"* />

## 8.4 一对多关联

collection 支持的属性以及属性的作用和 association 完全相同。mybatis会根据id标签，进行字段的合并，合理配置好ID标签可以提高处理的效率；

示例代码：com.enjoylearning.mybatis.MybatisDemo.testManyParamQuery()

开发小技巧：如果要配置一个相当复杂的映射，一定要从基础映射开始配置，每增加一些配置就进行对应的测试，在循序渐进的过程中更容易发现和解决问题 。

## 8.5 多对多关联

要实现多对多的关联，需要满足如下两个条件：

1. 先决条件一：多对多需要一种中间表建立连接关系；
2. 先决条件二：多对多关系是由两个一对多关系组成的，一对多可以也可以用两种方式实现；

示例代码：com.enjoylearning.mybatis.AssociationQueryTest.testManyToMany()

# 缓存

MyBatis 包含一个非常强大的查询缓存特性，使用缓存可以使应用更快地获取数据，避免频繁的数据库交互 ；

## 9.1一级缓存

一级缓存默认会启用，想要关闭一级缓存可以在select标签上配置flushCache=“true”；一级缓存存在于 **SqlSession 的生命周期**中，在同一个 SqlSession 中查询时， MyBatis 会把执行的方法和参数通过算法生成缓存的键值，将键值和查询结果存入一个 Map对象中。如果同一个 SqlSession 中执行的方法和参数完全一致，那么通过算法会生成相同的键值，当 Map 缓存对象中己经存在该键值时，则会返回缓存中的对象;任何的 INSERT 、UPDATE 、 DELETE 操作都会清空一级缓存；

示例代码：com.enjoylearning.mybatis.MybatisCacheTest.Test1LevelCache()

## 9.2二级缓存

二级缓存也叫应用缓存，存在于 **SqlSessionFactory 的生命周期**中，可以理解为跨sqlSession；缓存是以namespace为单位的，不同namespace下的操作互不影响。在MyBatis的核心配置文件中 cacheEnabled参数是二级缓存的全局开关，默认值是 true，如果把这个参数设置为 false，即使有后面的二级缓存配置，也不会生效；

要开启二级缓存,你需要在你的 SQL Mapper文件中添加配置：

<cache eviction=“LRU" flushInterval="60000" size="512" readOnly="true"/>

这段配置的效果如下：

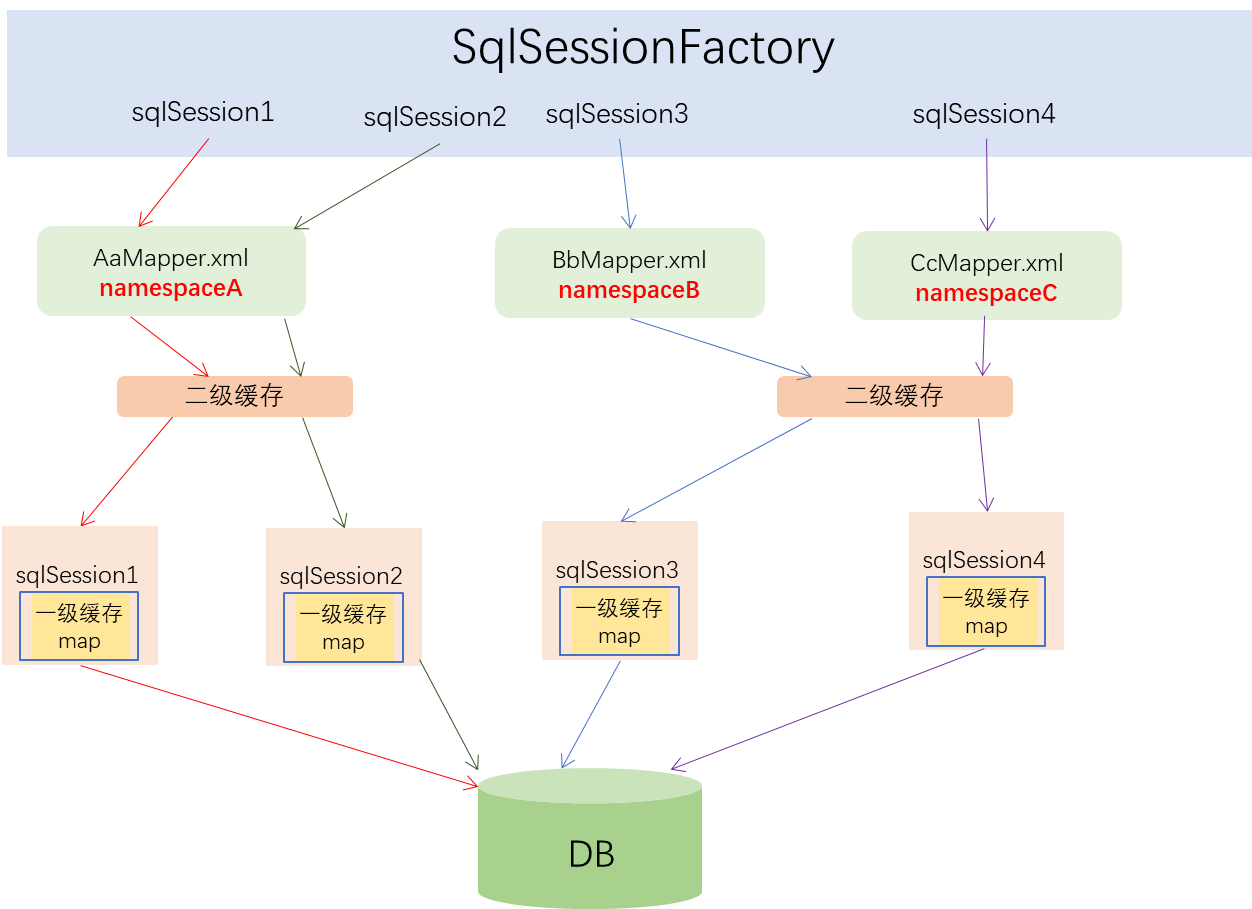
* 映射语句文件中的所有 select 语句将会被缓存。
* 映射语句文件中的所有 insert,update 和 delete 语句会刷新缓存。
* 缓存会使用 Least Recently Used(LRU,最近最少使用的)算法来收回。
* 根据时间表(比如 no Flush Interval,没有刷新间隔), 缓存不会以任何时间顺序 来刷新。
* 缓存会存储列表集合或对象(无论查询方法返回什么)的 512个引用。
* 缓存会被视为是 read/write(可读/可写)的缓存；

开发建议：使用二级缓存容易出现脏读，建议避免使用二级缓存，在业务层使用可控制的缓存代替更好；

示例代码：com.enjoylearning.mybatis.MybatisCacheTest.Test2LevelCache()

## 9.3缓存调用过程

缓存的调用过程如下：



调用过程解读：

1. 每次与数据库的连接都会优先从缓存中获取数据
2. 先查二级缓存，再查一级缓存
3. 二级缓存以namespace为单位的，是SqlSession共享的，容易出现脏读，建议避免使用二级缓存
4. 一级缓存是SqlSession独享的，建议开启；