专题35：mybatis面试题（史上最全、定期更新）

本文版本说明：V3.0

> 此文的格式，由markdown 通过程序转成而来，由于很多表格，没有来的及调整，出现一个格式问题，尼恩在此给大家道歉啦。
>
> 由于社群很多小伙伴，在面试，不断的交流最新的面试难题，所以，《Java面试红宝书》， 后面会不断升级，迭代。
>
> 本专题，作为 《Java面试红宝书》专题之一， 《Java面试红宝书》一共**30个面试专题，后续还会增加**

《尼恩面试宝典》升级的规划为：

后续基本上，**每一个月，都会发布一次**，最新版本，可以扫描扫架构师尼恩微信， 发送 “领取电子书” 获取。

尼恩的微信二维码在哪里呢？ 请参见文末

面试问题交流说明：

如果遇到面试难题，或者职业发展问题， 或者中年危机问题，都可以来 疯狂创客圈社群交流，

入交流群，加尼恩微信即可， 发送**“入群”**，

尼恩微信请打开语雀扫码 https://www.yuque.com/crazymakercircle/gkkw8s/khigna

mybatis面试题

聊聊：MyBatis是什么?

Mybatis 是一个半 ORM（对象关系映射）框架，它内部封装了 JDBC，开发时只需要关注 SQL 语句本身，不需要花费精力去处理加载驱动、创建连接、创建statement 等繁杂的过程。程序员直接编写原生态 sql，可以严格控制 sql 执行性能，灵活度高。

MyBatis 可以使用 XML 或注解来配置和映射原生信息，将 POJO 映射成数据库中的记录，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

聊聊：Mybatis优缺点

优点

与传统的数据库访问技术相比，ORM有以下优点：

- 基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在

XML里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用

- 与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接

很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据库，所以只要JDBC支持的数据库MyBatis都支持）

- 提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护
- 能够与Spring很好的集成

缺点

- SQL语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写SQL语句的功底有一定要求
- SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库

聊聊：为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？

- Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。
- 而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。

聊聊：传统JDBC开发存在什么问题？

- 频繁创建数据库连接对象、释放，容易造成系统资源浪费，影响系统性能。

  可以使用连接池解决这 个问题。但是使用jdbc需要自己实现连接池。
- sql语句定义、参数设置、结果集处理存在硬编码。

  实际项目中sql语句变化的可能性较大，一旦发 生变化，需要修改java代码，系统需要重新编译，重新发布。不好维护。
- 使用preparedStatement向占有位符号传参数存在硬编码，

  因为sql语句的where条件不一定，可能多也可能少，修改sql还要修改代码，系统不易维护。
- 结果集处理存在重复代码，处理麻烦。

  如果可以映射成Java对象会比较方便。

聊聊：MyBatis 的好处是什么？

答:

1）MyBatis 把 sql 语句从 Java 源程序中独立出来，放在单独的 XML 文件中编写，给程序的 维护带来了很大便利。

2）MyBatis 封装了底层 JDBC API 的调用细节，并能自动将结果集转换成 Java Bean 对象， 大大简化了 Java 数据库编程的重复工作。

3）因为 MyBatis 需要程序员自己去编写 sql 语句，程序员可以结合数据库自身的特点灵活 控制 sql 语句，因此能够实现比 Hibernate 等全自动 orm 框架更高的查询效率，能够完成复 杂查询。

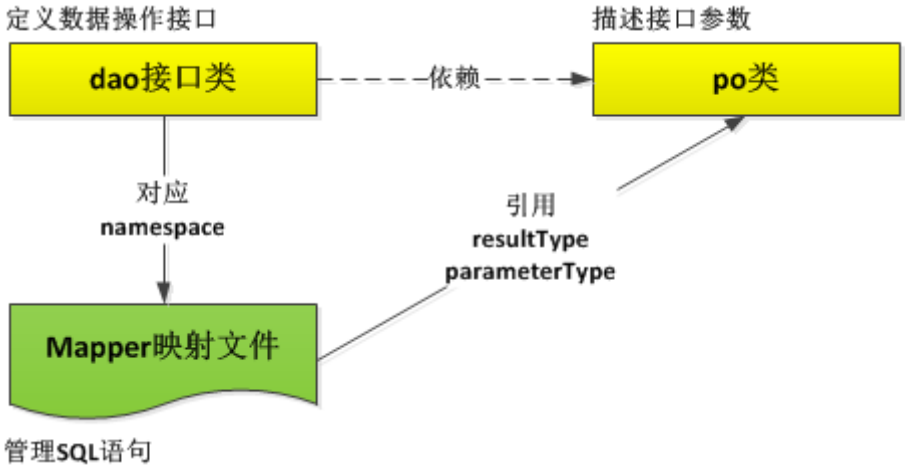
基础：MyBatis之XML映射文件详解

MyBatis 的真正强大在于它的映射语句，也是它的魔力所在。

由于它的异常强大，映射器的 XML 文件就显得相对简单。

在MyBatis开发中，涉及到主要开发要素是：Dao接口类，Mapper映射文件，以及PO类。

它们之间的关系如下：



一个Mapper映射文件的例子

```
<?xml version="1.0" encoding="UTF8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mb.dao.StudentMapper">
    <!-- 配置方式一：通过结果集映射的方式进行查询 -->
    <select id="getStudentList" resultMap="StudentTeacherMap">
        select s.id sid, s.name sname, t.name tname
        from student s, teacher t
        where s.tid = t.id;
    </select>

  <resultMap id="StudentTeacherMap" type="Student">
        <result property="id" column="sid"/>
        <result property="name" column="sname"/>
        <!-- association表示关联对象：javaType表示关联对象的java类型，
        因为关联对象只有一个，而不是集合，所以直接配置关联对象对应属性即可-->
        <association property="teacher" javaType="Teacher">
            <!-- 因为结果集中没有teacher的id，所以这里就不用配置id了 -->
            <result property="name" column="tname"/>
```

```
        </association>
    </resultMap>

    <!-- 配置方式二：通过子查询的方式进行查询，这种方式的缺点是，因为每个查询都要单独配置，所以
不能直接使用完整的SQL去调试 -->
    <select id="getStudentList2" resultMap="StudentTeacherMap2">
        select * from student;
    </select>
    <select id="getTeacherById" resultType="Teacher">
        select * from teacher where id = #{tid};
    </select>
    <resultMap id="StudentTeacherMap2" type="Student">
        <result property="id" column="id"/>
        <result property="name" column="id"/>
        <!-- association表示关联对象： column表示要传入子查询的字段，javaType表示关联对
象的java类型，
        select表示子查询，这里表示将查询到的column="tid"传入子查询
select="getTeacherById"作为参数#{tid}的值进行查询 -->
        <association property="teacher" column="tid" javaType="Teacher"
select="getTeacherById"/>
    </resultMap>
</mapper>
```

Mapper映射文件的顶级元素

映射器（mapper）的XML文件，有几个顶级元素：

- select – 映射查询语句
- insert – 映射插入语句
- update – 映射更新语句
- delete – 映射删除语句
- sql – 可被其他语句引用的可重用语句块。
- 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。

1、select元素

1. 最基本的查询

```
    <select id="getUserById" resultType="MemberUser" parameterType="int">
    select ID,NAME,PERSONMOBILE,ADDRESS,AGE FROM MEMBER_USER WHERE ID = #{id}
    </select>
```

上述配置类似于：

```
    // Similar JDBC code, NOT MyBatis…
    String selectMember = "select ID,NAME,PERSONMOBILE,ADDRESS,AGE FROM
MEMBER_USER WHERE ID=?";
    PreparedStatement ps = conn.prepareStatement(selectMember);
    ps.setInt(1,id);
```

2. select 元素有很多属性允许你配置，来决定每条语句的作用细节

示范：

```
<select
  id="selectPerson"
  parameterType="int"
  parameterMap="deprecated"
  resultType="hashmap"
  resultMap="personResultMap"
  flushCache="false"
  useCache="true"
  timeout="10000"
  fetchSize="256"
  statementType="PREPARED"
  resultSetType="FORWARD_ONLY">
```

详细说明：

属性	描述
id	在命名空间中唯一的标识符，可以被用来引用这条语句。
parameterType	将会传入这条语句的参数类的完全限定名或别名。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。
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。
resultMap	外部 resultMap 的命名引用。结果集的映射是 MyBatis 最强大的特性，对其有一个很好的理解的话，许多复杂映射的情形都能迎刃而解。使用 resultMap 或 resultType，但不能同时使用。
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：false。
useCache	将其设置为 true，将会导致本条语句的结果被二级缓存，默认值：对 select 元素为 true。
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）。
fetchSize	这是尝试影响驱动程序每次批量返回的结果行数和这个设置值相等。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。
resultSetType	FORWARD_ONLY，SCROLL_SENSITIVE 或 SCROLL_INSENSITIVE 中的一个，默认值为 unset （依赖驱动）。
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。
resultOrdered	这个设置仅针对嵌套结果 select 语句适用：如果为 true，就是假设包含了嵌套结果集或是分组了，这样的话当返回一个主结果行的时候，就不会发生有对前面结果集的引用的情况。这就使得在获取嵌套的结果集的时候不至于导致内存不够用。默认值：false。
resultSets	这个设置仅对多结果集的情况适用，它将列出语句执行后返回的结果集并每个结果集给一个名称，名称是逗号分隔的。

2、insert、update、delete元素

1. 数据变更语句 insert，update 和 delete 的实现非常接近，参考如下配置：

```
    <!--Oracle的实现自增长主键的方式-->
    <insert id="insertUser" parameterType="MemberUser">
    <selectKey keyProperty="id" resultType="int" order="BEFORE">
      select SEQ_MEMBER_USER.nextval from DUAL
```

```
    </selectKey>
      INSERT INTO MEMBER_USER (ID, NAME, PERSONMOBILE, ADDRESS, AGE)
      VALUES(#{id}, #{name}, #{personMobile}, #{address}, #{age})
    </insert>

    <update id="updateUser" parameterType="MemberUser">
      update MEMBER_USER set
        NAME = #{name},
        PERSONMOBILE = #{personMobile},
        ADDRESS = #{address},
        AGE = #{age}
      where id = #{id}
    </update>

    <delete id="deleteUser" parameterType="int">
      delete from MEMBER_USER where ID = #{id}
    </delete>
```

2 .Insert, Update 和 Delete 的属性:

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。

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。
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。

3. 关于insert元素

如果你的数据库支持自动生成主键的字段（比如 MySQL 和 SQL Server），那么你可以设置 useGeneratedKeys=”true”，然后再把 keyProperty 设置到目标属性上就OK了。示范：

```
<insert id="insertAuthor" useGeneratedKeys="true"
    keyProperty="id">
  insert into Author (username,password,email,bio)
  values (#{username},#{password},#{email},#{bio})
</insert>
```

如果是Oracle数据库，则用上面示范代码即可。selectKey 元素描述如下：

```
<selectKey
  keyProperty="id"
  resultType="int"
  order="BEFORE"
  statementType="PREPARED">
```

selectKey 的属性：

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。

3、sql元素

这个元素可以被用来定义可重用的 SQL 代码段，可以包含在其他语句中。

1. 简单示范：

```
    <!-- 用来定义可重用的SQL代码段 -->
    <sql id="selectProdSQL">

PRODID,PRODSERIAL,PRODNAME,CATEGORYNAME,PRODSPEC,PRODPRICE,PRODDESC,PRODIMAGE,IS
NEW,ISRECOMMEND,ISSHOW,USERNAME,HANDLETIME,PRODTHUMBNAIL
    </sql>

    <!--以单个对象方式返回-->
    <select id="getProductById" resultType="Product" parameterType="int">
    select <include refid="selectProdSQL"/>
      FROM PRODUCT
    WHERE PRODID = #{prodId}
    </select>
```

2. 进阶用法：

```
<sql id="userColumns"> ${alias}.id,${alias}.username,${alias}.password </sql>
```

这个 SQL 片段可以被包含在其他语句中，例如：

```
<select id="selectUsers" resultType="map">
  select
    <include refid="userColumns"><property name="alias" value="t1"/></include>,
    <include refid="userColumns"><property name="alias" value="t2"/></include>
  from some_table t1
    cross join some_table t2
</select>
```

4、resultMap元素

ResultMap 的设计就是简单语句不需要明确的结果映射,而很多复杂语句确实需要描述它们 的关系。
resultmap构成元素：

1. id:

>> 一般对应数据库中改行的主键ID，设置此项可以提高Mybatis的性能

2. result

>> 映射到JavaBean的某个“简单类型”属性

3. association

>> 映射到javaBean的某个“复杂类型”属性，比如：javabean类

4. collection

>> 映射到javabean的某个“复杂类型”属性，比如：集合

```
<!--column不做限制，可以为任意表的字段，而property须为type 定义的pojo属性-->
<resultMap id="唯一的标识" type="映射的pojo对象">
  <id column="表的主键字段，或者可以为查询语句中的别名字段" jdbcType="字段类型"
property="映射pojo对象的主键属性" />
  <result column="表的一个字段（可以为任意表的一个字段）" jdbcType="字段类型"
property="映射到pojo对象的一个属性（须为type定义的pojo对象中的一个属性）"/>
  <association property="pojo的一个对象属性" javaType="pojo关联的pojo对象">
    <id column="关联pojo对象对应表的主键字段" jdbcType="字段类型" property="关联pojo对
象的主席属性"/>
    <result  column="任意表的字段" jdbcType="字段类型" property="关联pojo对象的属性"/>
  </association>
  <!-- 集合中的property须为oftype定义的pojo对象的属性-->
  <collection property="pojo的集合属性" ofType="集合中的pojo对象">
    <id column="集合中pojo对象对应的表的主键字段" jdbcType="字段类型" property="集合中
pojo对象的主键属性" />
    <result column="可以为任意表的字段" jdbcType="字段类型" property="集合中的pojo对象
的属性" />
  </collection>
</resultMap>
```

正常情况，它用来作为将数据库字段与PO类的字段进行映射，比如：

```
<resultMap id="BaseResultMap" type="com.meikai.shop.entity.TShopSku">
    <id column="ID" jdbcType="BIGINT" property="id" />
    <result column="SKU_NAME" jdbcType="VARCHAR" property="skuName" />
    <result column="CATEGORY_ID" jdbcType="BIGINT" property="categoryId" />
</resultMap>
```

基础：OGNL 表达式（Object-Graph Navigation Language 对象-图形导航语言）

OGNL是Object-Graph Navigation Language(对象图导航语言)的缩写，

它是一种功能强大的表达式语言。strusts/MyBatis大量使用了 OGNL表达式。

OGNL 表达式的作用：可以`存取对象的属性`和调用对象的方法，通过OGNL 表达式可以迭代获取对象的结构图

MyBatis中使用了OGNL，MyBatis中可以使用OGNL的地方有两处：

- 动态SQL表达式中
- ${param}参数中

上面这两处地方在MyBatis中处理的时候都是使用OGNL处理的。

MyBatis常用OGNL表达式

- e1 or e2
- e1 and e2
- e1 == e2,e1 eq e2
- e1 != e2,e1 neq e2
- e1 lt e2：小于
- e1 lte e2：小于等于，其他gt（大于）,gte（大于等于）
- e1 in e2
- e1 not in e2
- e1 + e2,e1 * e2,e1/e2,e1 - e2,e1%e2
- !e,not e：非，求反
- e.method(args)调用对象方法
- e.property对象属性值
- e1[e2]按索引取值，List,数组和Map
- @class@method(args)调用类的静态方法
- @class@field调用类的静态字段值

上述内容只是合适在MyBatis中使用的OGNL表达式，完整的表达式点击这里。

1、语法：#{ }

```
#{ }:  是指上下文(环境)对象
```

OGNL 表达式获取属性举例：

对象person:{id:10,age:18,name:小明};

若上下文(环境)的对象是person，通过#{ } 可以直接获取到对象的属性值

```
#{id} 相当于 perosn.getId()
#{age} 相当于 person.getAge()
#{name} 相当于 person.getName()
```

OGNL 在mybatis框架中的应用：

```
/* User 类*/
@Data
public class User {
    private Long id;
    private String name;
    private BigDecimal salary;
}

/* 测试类 */
User user = new User();
user.setId(2L);
SqlSession session = factory.openSession();
//4、进行数据库操作（CRUD）
User user = session.selectOne("com.shan.hello.UserMapper.get", user);//将user作为上下文对象(javaBean类型)传入

/* 映射文件 */
<select id="get" parameterType="java.lang.Long"
resultType="com.shan.hello.User">
    select * from t_user where id = #{id}   //上下文是javaBean类型，OGNL 表达式格式必须为 #{属性名}
</select>
```

OGNL 的上下文对象类型【获取属性值】：

（1）javaBean对象，例如上面的User，则 OGNL 表达式格式必须为 #{属性名}

（2）map 对象，OGNL 表达式格式为 #{key}

（3）简单类型对象（基本类型、String类型），OGNL 表达式格式为 #{随便写，不过一般写得见名知意，增加代码的阅读性}

简单类型举例子：

```
/* 测试类 */
SqlSession session = factory.openSession();
//4、进行数据库操作（CRUD）
User user = session.selectOne("com.shan.hello.UserMapper.get", 2L);//将2L作为上下文对象(简单类型)传入

/* 映射文件 */
<select id="get" parameterType="java.lang.Long"
resultType="com.shan.hello.User">
    select * from t_user where id = #{id}//上下文是javaBean类型，OGNL 表达式格式可以写成#{aa}，不过增强代码阅读性也会写成#{id}
</select>
```

EL 表达式 ${ }

1、语法：${属性名}

2、OGNL 表达式的作用：通过 ${属性名} 直接获取属性值，属性的内容。

在MyBatis中，OGNL 表达式 #{} 和 EL 表达式 ${} 的异同

1、sql语句分别使用#{} 和 ${}的实际情况：

（1）使用 #{}：

```
<!-- 映射文件 --->
<select id="login" resultType="Client">
    select id, username, password from client where username = #{username} and password = #{password} ;
 </select>

 <!-- #{} 的实际作用：现在转成?的占位符，然后再把值设置进去【假设外界传入的值username="小明"，password="1"】 -->
select id, username, password from client where username = ? and password = ?;
PreparedStatement.setString(1, "小明");
PreparedStatement.setString(2,"1");
```

（2）使用 ${}：

```
<!-- 映射文件 --->
<select id="login" resultType="Client">
    select id, username, password from client where username = ${username} and
password = ${password} ;
 </select>

 <!-- ${} 的实际作用：直接把值设置进去【假设外界传入的值username="小明"，password="1"】 -
->
select id, username, password from client where username = "小明" and password =
"1";
```

2、# 和 $ 的异同：

- 相同：都可以获取对象的信息。
- 不同：

  使用# 传递的参数，会先转成占位符？，再通过设置占位符参数的方式设置值【会给值用单引号引起来】，不会导致sql注入问题，比较安全。

  使用$ 传递的参数，直接把解析出来的数据作为sql语句的一部分。可能会出现sql注入安全问题，比较不安全。

聊聊：{}和${}的区别

- {}是占位符，预编译处理；${}是拼接符，字符串替换，没有预编译处理。
- Mybatis在处理#{}时，#{}传入参数是以字符串传入，会将SQL中的#{}替换为?号，调用PreparedStatement的set方法来赋值。
- #{} 可以有效的防止SQL注入，提高系统安全性；${} 不能防止SQL 注入
- #{} 的变量替换是在DBMS 中；${} 的变量替换是在 DBMS 外

聊聊：#{}和${}的区别是什么?

答：

1）#{}是预编译处理，${}是字符串替换。

2）Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的 set 方法来赋值；

3）Mybatis 在处理${}时，就是把${}替换成变量的值。

4）使用#{}可以有效的防止 SQL 注入，提高系统安全性。

聊聊：模糊查询like语句该怎么写

1 '%${question}%' 可能引起SQL注入，不推荐

2 "%"#{question}"%" 注意：

因为#{...}解析成sql语句时候，会在变量外侧自动加单引号' '，所 以 这里 % 需要使用双引号" "，不能使用单引号 ' '，不然会查不到任何结果。

3 CONCAT('%',#{question},'%') 使用CONCAT()函数，（推荐）

4 使用bind标签（不推荐）

```
<select id="listUserLikeUsername" resultType="com.jourwon.pojo.User">

    <bind name="pattern" value="'%' + username + '%'" />

    select id,sex,age,username,password from person where username LIKE {pattern}

</select>
```

聊聊：在mapper中如何传递多个参数

方法1：顺序传参法

```
public User selectUser(String name, int deptId);

<select id="selectUser" resultMap="UserResultMap">

select * from user where user_name = #{0} and dept_id = #{1}

</select>
```

- #{}里面的数字代表传入参数的顺序。
- 这种方法不建议使用，sql层表达不直观，且一旦顺序调整容易出错。

方法2：@Param注解传参法

```
public User selectUser(@Param("userName") String name, int @Param("deptId")

deptId);

<select id="selectUser" resultMap="UserResultMap">

select * from user

where user_name = #{userName} and dept_id = #{deptId}

</select>
```

- #{}里面的名称对应的是注解@Param括号里面修饰的名称。
- 这种方法在参数不多的情况还是比较直观的，（推荐使用）。

方法3：Map传参法

```
public User selectUser(Map<String, Object> params);

<select id="selectUser" parameterType="java.util.Map" resultMap="UserResultMap">

select * from user

where user_name = #{userName} and dept_id = #{deptId}

</select>
```

- #{}里面的名称对应的是Map里面的key名称。
- 这种方法适合传递多个参数，且参数易变能灵活传递的情况。（推荐使用）。

方法4：Java Bean传参法

```
public User selectUser(User user);

<select id="selectUser" parameterType="com.jourwon.pojo.User"

resultMap="UserResultMap">

select * from user

where user_name = #{userName} and dept_id = #{deptId}

</select>
```

- #{}里面的名称对应的是User类里面的成员属性。
- 这种方法直观，需要建一个实体类，扩展不容易，需要加属性，但代码可读性强，业务逻辑处理方便，推荐使用。（推荐使用）。

聊聊：Mybatis是如何将sql执行结果封装为目标对象并返回的？都有哪 些映射形式?

- 第一种是使用 标签，逐一定义列名和对象属性名之间的映射关系。

- 第二种是使用sql列的别名功能，将列别名书写为对象属性名

  比如T_NAME AS NAME，对象属性 名一般是name，小写，但是列名不区分大小写，Mybatis会忽略列名大小写，智能找到与之对应对象属性名，你甚至可以写成T_NAME AS NaMe，Mybatis一样可以正常工作。

  有了列名与属性名的映射关系后，Mybatis通过反射创建对象，同时使用反射给对象的属性逐一赋值并返回，那 些找不到映射关系的属性，是无法完成赋值的

聊聊：Xml映射文件中，除了常见的select|insert|updae|delete标 签之外，还有哪些标签?

还有很多其他的标签，

、（被弃用）、 、 、 ，

加上动态sql的9个标签

trim|where|set|foreach|if|choose|when|otherwise|bind等，

其中 为sql片段标签，通 过 标签引入sql片段， 为不支持自增的主键生成策略标签

聊聊：Mybatis 是如何将 sql 执行结果封装为目标对象并返回的?都有哪些映射形式?

第一种是使用标签，逐一定义列名和对象属性名之间的映射关系。

第二 种是使用 sql 列的别名功能，将列别名书写为对象属性名，比如 T_NAME AS NAME，对象属性名一般是 name，小写，但是列名不区分大小写，Mybatis 会忽略列名大小 写，智能找到与之对应对象属性名，

你甚至可以写成 T_NAME AS NaMe，Mybatis 一样可以正常工作。

有了列名与属性名的映射关系后，Mybatis 通过反射创建对象，同时使用反射给对象 的属性逐一赋值并返回，那些找不到映射关系的属性，是无法完成赋值的。

聊聊：Mybatis动态sql是做什么的？都有哪些动态sql？能简述一下动态sql的执行原理吗?

- Mybatis动态sql可以让我们在Xml映射文件内，以标签的形式编写动态sql，完成逻辑判断和动态拼接sql的功能，

  Mybatis提供了9种动态sql标签 trim|where|set|foreach|if|choose|when|otherwise|bind。
- 其执行原理为，使用OGNL从sql参数对象中计算表达式的值，根据表达式的值动态拼接sql，以此来完成动态sql的功能。

聊聊： 一级、二级缓存

1）一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有 Cache 就将 清空。

2）二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存 储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如Ehcache。

要开启二级缓存，你需要在你的 SQL 映射文件中添加一行:

3）对于缓存数据更新机制，当某一个作用域(一级缓存Session/二级缓存Namespaces)的进行了 C/U/D 操作后，默认该作用域下所有 select 中的缓存将 被 clear。

聊聊：Mybatis 是否支持延迟加载？如果支持，它的实现原理是什么？

Mybatis 仅支持 association 关联对象和 collection 关联集合对象的延迟加载，

’association 指的就是一对一，collection 指的就是一对多查询。

在 Mybatis 配置文 件中，可以配置是否启用延迟加载 lazyLoadingEnabled=true|false。

它的原理是，使用 CGLIB 创建目标对象的代理对象，当调用目标方法时，进入拦截 器方法，比如调用 a.getB().getName()，拦截器 invoke()方法发现 a.getB()是 null 值，

那么就会单独发送事先保存好的查询关联 B 对象的 sql，把 B 查询上来，然后调 用 a.setB(b)，于是 a 的对象 b 属性就有值了，接着完成 a.getB().getName()方法的 调用。

这就是延迟加载的基本原理。

聊聊：Mybatis 映射文件中，如果 A 标签通过 include 引用了 B 标签的内容，请问，B 标签能否定义在 A 标签的后面，还是说必须定义在 A 标签的前面？

虽然 Mybatis 解析 Xml 映射文件是按照顺序解析的，但是，被引用的 B 标签依然可以定义在任何地方，Mybatis 都可以正确识别。

原理是，Mybatis 解析 A 标签，发现 A 标签引用了 B 标签，但是 B 标签尚未解析到， 尚不存在，

此时，Mybatis 会将 A 标签标记为未解析状态，然后继续解析余下的标签， 包含 B 标签，待所有标签解析完毕，Mybatis 会重新解析那些被标记为未解析的标 签，

此时再解析 A 标签时，B 标签已经存在，A 标签也就可以正常解析完成了。

聊聊： Mybatis 的 Xml 映射文件和 Mybatis 内部数据结构之间的映射关系？

Mybatis 将所有 Xml 配置信息都封装到 All-In-One 重量级对象 Configuration 内 部。

在 Xml 映射文件中，标签会被解析为 ParameterMap 对象，

其每个子元素会被解析为 ParameterMapping 对象。标签会被解析为 ResultMap 对象，其每个子元素会被解析为 ResultMapping 对象。

每一个<select>、 <insert>、<update>、<delete>标签均会被解析为 MappedStatement 对象，

标签 内的 sql 会被解析为 BoundSql 对象。

聊聊：接口绑定有几种实现方式,分别是怎么实现的?

答：接口绑定有两种实现方式,一种是通过注解绑定,就是在接口的方法上面加上 @Select@Update 等注解里面包含 Sql 语句来绑定,另外一种就是通过 xml 里面写 SQL 来绑 定,在这种情况下,要指定 xml 映射文件里面的 namespace 必须为接口的全路径名.

聊聊：什么情况下用注解绑定,什么情况下用 xml 绑定?

答：当 Sql 语句比较简单时候,用注解绑定；当 SQL 语句比较复杂时候,用 xml 绑定,一般用 xml 绑定的比较多

聊聊：MyBatis 实现一对一有几种方式?具体怎么操作的?

答：有联合查询和嵌套查询,

联合查询是几个表联合查询,只查询一次,通过在 resultMap 里面 配置 association 节点配置一对一的类就可以完成;

嵌套查询是先查一个表,根据这个表里面 的结果的外键 id,去再另外一个表里面查询数据,也是通过 association 配置,但另外一个表的 查询通过 select 属性配置。

聊聊：MyBatis 里面的动态 Sql 是怎么设定的?用什么语法?

答:

MyBatis 里面的动态 Sql 一般是通过 if 节点来实现,

通过 OGNL 语法来实现,但是如果要 写的完整,必须配合 where,trim 节点,where 节点是判断包含节点有内容就插入 where,

否则不 插入,trim 节点是用来判断如果动态语句是以 and 或 or 开始,那么会自动把这个 and 或者 or 取掉。

聊聊：Mybatis 是如何将 sql 执行结果封装为目标对象并返回的？都有哪些映射形式?

答:

第一种是使用标签，逐一定义列名和对象属性名之间的映射关系。

第二种是使用 sql 列的别名功能，将列别名书写为对象属性名，比如 T_NAME AS NAME，对 象属性名一般是 name，小写，但是列名不区分大小写，Mybatis 会忽略列名大小写，智能找到与之对应对象属性名，你甚至可以写成 T_NAME AS NaMe，Mybatis 一样可以正常工 作。

有了列名与属性名的映射关系后，Mybatis 通过反射创建对象，同时使用反射给对象的属性逐一赋值并返回，那些找不到映射关系的属性，是无法完成赋值的。

聊聊：当实体类中的属性名和表中的字段名不一样，如果将查询的结果封装到指定 pojo?

答:

1）通过在查询的 sql 语句中定义字段名的别名。

2）通过来映射字段名和实体类属性名的一一对应的关系。

聊聊：通常一个 Xml 映射文件，都会写一个 Dao 接口与之对应，Dao 的工作原理，是否可以重

载?

答：不能重载，因为通过 Dao 寻找 Xml 对应的 sql 的时候全限名+方法名的保存和寻找策

略。接口工作原理为 jdk 动态代理原理，运行时会为 dao 生成 proxy，代理对象会拦截接口

方法，去执行对应的 sql 返回数据。

聊聊：Mybatis 的 Xml 映射文件中，不同的 Xml 映射文件，id 是否可以重复?

答:

不同的 Xml 映射文件，如果配置了 namespace，那么 id 可以重复；

如果没有配置 namespace，那么 id 不能重复；毕竟 namespace 不是必须的，只是最佳实践而已。

原因就 是 namespace+id 是作为 Map<String, MappedStatement>的 key 使用的，

如果没有 namespace，就剩下 id，那么，id 重复会导致数据互相覆盖。

有了 namespace，自然 id 就 可以重复，namespace 不同，namespace+id 自然也就不同。

聊聊：Mybatis 都有哪些 Executor 执行器？它们之间的区别是什么?

答:

Mybatis 有三种基本的 Executor 执行器，SimpleExecutor、ReuseExecutor、 BatchExecutor。

1）SimpleExecutor：每执行一次 update 或 select，就开启一个 Statement 对 象，用完立刻关闭 Statement 对象。

2）ReuseExecutor：执行 update 或 select，以 sql 作为 key 查找 Statement 对象，存在就使用，不存在就创建，用完后，不关闭 Statement 对象， 而是放置于 Map

3）BatchExecutor：完成批处理。

聊聊：Mybatis 中如何指定使用哪一种 Executor 执行器?

答:

在 Mybatis 配置文件中，可以指定默认的 ExecutorType 执行器类型，也可以手动给 DefaultSqlSessionFactory 的创建 SqlSession 的方法传递 ExecutorType 类型参数。

聊聊：Mybatis 执行批量插入，能返回数据库主键列表吗?

答：能，JDBC 都能，Mybatis 当然也能。

聊聊：如何获取自动生成的(主)键值?

答：配置文件设置 usegeneratedkeys 为 true

聊聊：Mybatis 是如何进行分页的？分页插件的原理是什么?

答:

1）Mybatis 使用 RowBounds 对象进行分页，也可以直接编写 sql 实现分页，也可以使用 Mybatis 的分页插件。

2）分页插件的原理：实现 Mybatis 提供的接口，实现自定义插件，在插件的拦截方法内拦 截待执行的 sql，然后重写 sql。

举例:

select * from student，

拦截 sql 后重写为:

select t.* from （select * from student）t limit 0，10

聊聊：Mybatis 动态 sql 是做什么的？都有哪些动态 sql？能简述一下动态 sql 的执行原理不?

答:

1）Mybatis 动态 sql 可以让我们在 Xml 映射文件内，以标签的形式编写动态 sql，完成逻辑 判断和动态拼接 sql 的功能。

2）Mybatis 提供了 9 种动态 sql 标签：

trim|where|set|foreach|if|choose|when|otherwise|bind。

3）其执行原理为，使用 OGNL 从 sql 参数对象中计算表达式的值，根据表达式的值动态拼 接 sql，以此来完成动态 sql 的功能。

聊聊：MyBatis 的好处是什么?

答：

1）MyBatis 把 sql 语句从 Java 源程序中独立出来，放在单独的 XML 文件中编写，给程序的 维护带来了很大便利。

2）MyBatis 封装了底层 JDBC API 的调用细节，并能自动将结果集转换成 Java Bean 对象， 大大简化了 Java 数据库编程的重复工作。

3）因为 MyBatis 需要程序员自己去编写 sql 语句，程序员可以结合数据库自身的特点灵活 控制 sql 语句，因此能够实现比 Hibernate 等全自动 orm 框架更高的查询效率，能够完成复 杂查询。

聊聊：什么情况下用注解绑定,什么情况下用 xml 绑定?

答：当 Sql 语句比较简单时候,用注解绑定；当 SQL 语句比较复杂时候,用 xml 绑定,一般用 xml 绑定的比较多

聊聊：MyBatis 里面的动态 Sql 是怎么设定的?用什么语法?

答：

MyBatis 里面的动态 Sql 一般是通过 if 节点来实现,通过 OGNL 语法来实现,

但是如果要 写的完整,必须配合 where,trim 节点,

where 节点是判断包含节点有内容就插入 where,否则不 插入,

trim 节点是用来判断如果动态语句是以 and 或 or 开始,那么会自动把这个 and 或者 or 取掉。

聊聊：Mybatis 的 Xml 映射文件中，不同的 Xml 映射文件，id 是否可以重复?

答：

不同的 Xml 映射文件，如果配置了 namespace，那么 id 可以重复；

如果没有配置 namespace，那么 id 不能重复；毕竟 namespace 不是必须的，只是最佳实践而已。

原因就 是 namespace+id 是作为 Map<String, MappedStatement>的 key 使用的，

如果没有 namespace，就剩下 id，那么，id 重复会导致数据互相覆盖。

有了 namespace，自然 id 就 可以重复，namespace 不同，namespace+id 自然也就不同。

聊聊：resultType resultMap 的区别?

答：

1）类的名字和数据库相同时，可以直接设置 resultType 参数为 Pojo 类

2）若不同，需要设置 resultMap 将结果名字和 Pojo 名字进行转换

聊聊：使用 MyBatis 的 mapper 接口调用时有哪些要求?

答：

1）Mapper 接口方法名和 mapper.xml 中定义的每个 sql 的 id 相同

2）Mapper 接口方法的输入参数类型和 mapper.xml 中定义的每个 sql 的 parameterType 的 类型相同

3）Mapper 接口方法的输出参数类型和 mapper.xml 中定义的每个 sql 的 resultType 的类型 相同

4）Mapper.xml 文件中的 namespace 即是 mapper 接口的类路径。

参考文献

https://www.cnblogs.com/nayitian/p/15075090.html

https://blog.csdn.net/weixin_45630258/article/details/122784096

https://www.cnblogs.com/wolf-lifeng/p/11153569.html

https://www.w3cschool.cn/mybatis/f4uw1ilx.html

https://www.cnblogs.com/kenhome/p/7764398.html

https://www.cnblogs.com/Charles-Yuan/p/9900279.html

硬核推荐：尼恩Java硬核架构班

又名疯狂创客圈社群 VIP

详情：

https://www.cnblogs.com/crazymakercircle/p/9904544.html

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的:

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

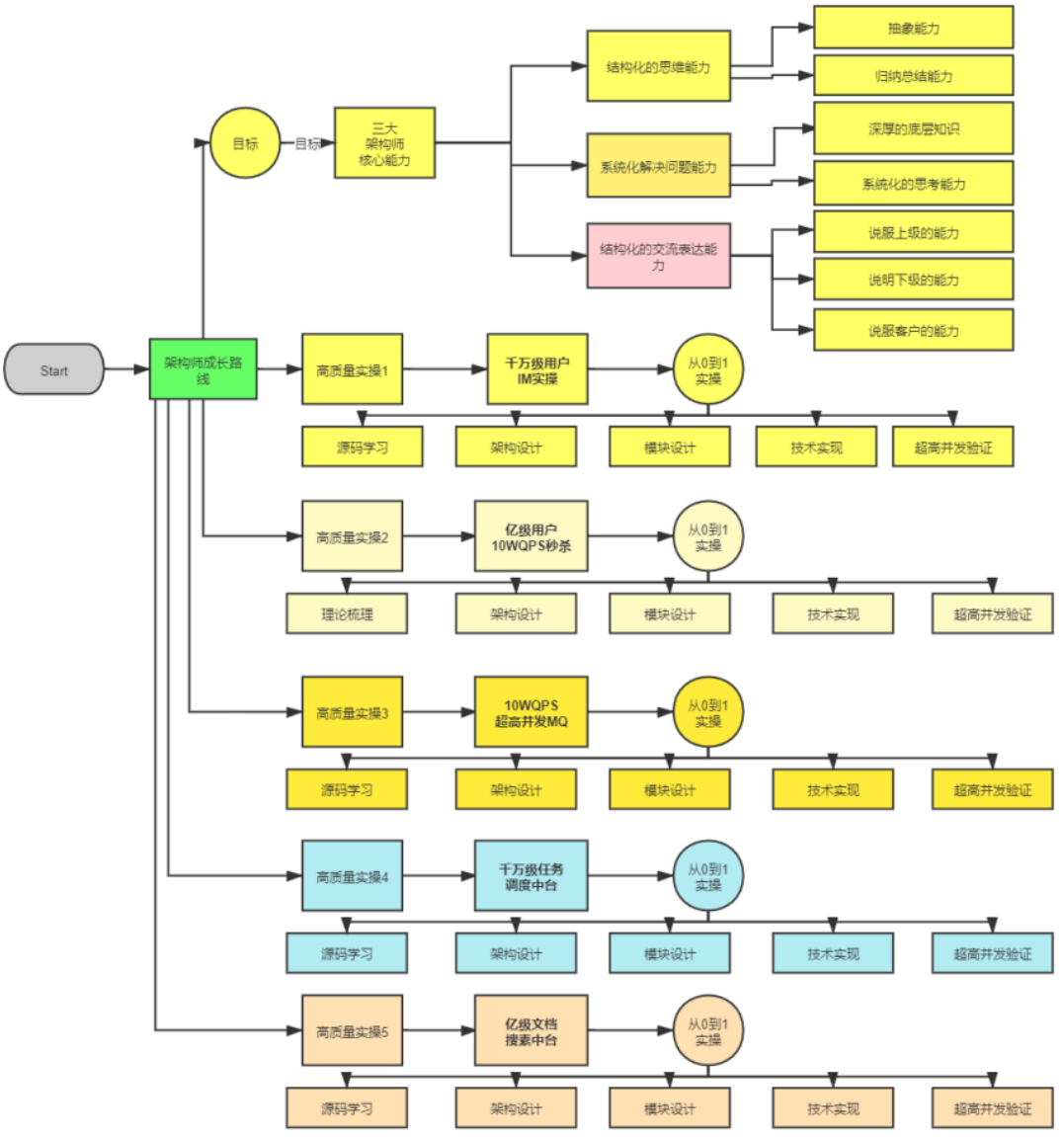
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章:横扫全网 Rocketmq 视频第 2 部曲: 工业级 rocketmq 高可用(HA)底层原理和实操

工业级 rocketmq 高可用底层原理，包含：消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现；消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操，包含：高可用集群的搭建。

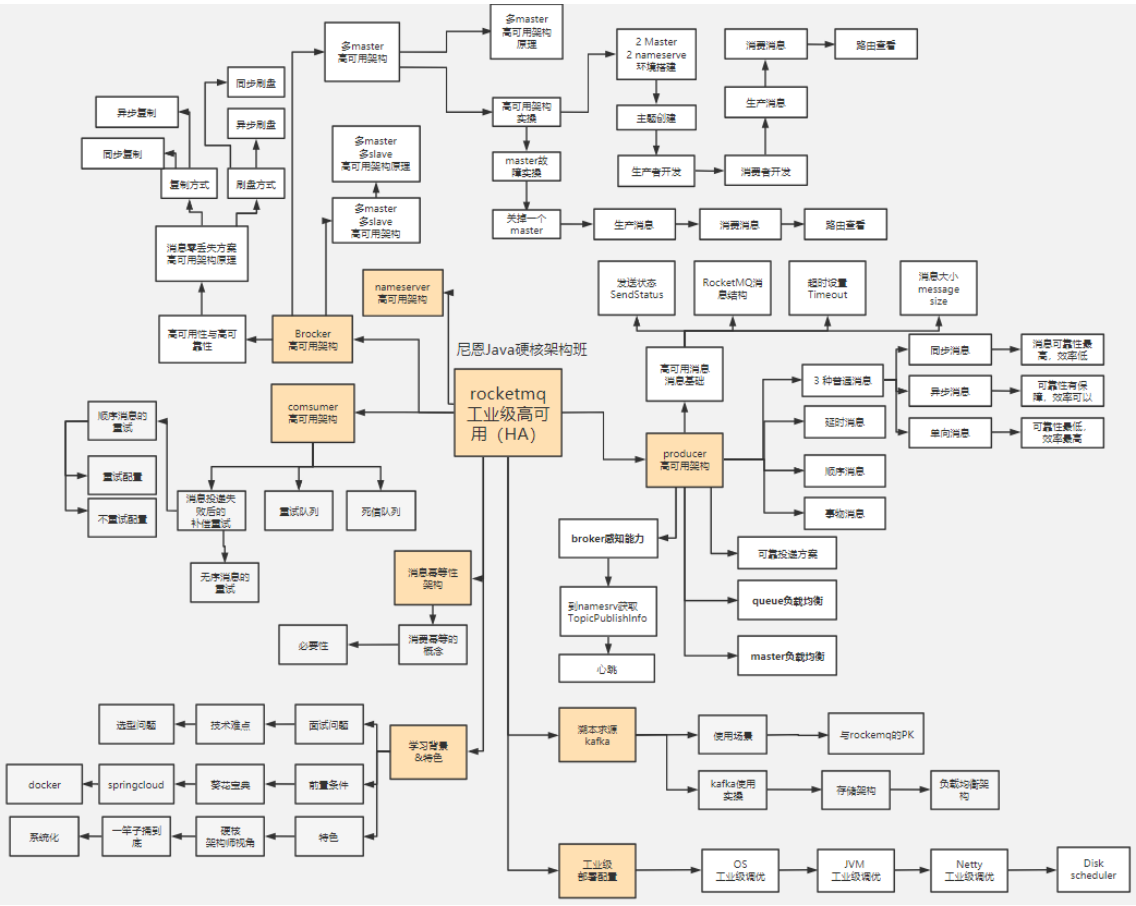
解决以下难题：

1、技术难题：RocketMQ 如何最大限度的保证消息不丢失的呢？RocketMQ 消息如何做到高可靠投递？

2、技术难题：基于消息的分布式事务，核心原理不理解

3、选型难题： kafka or rocketmq ，该娶谁？

下图链接：https://www.processon.com/view/6178e8ae0e3e7416bde9da19



成功案例：2 年翻 3 倍，35 岁卷王成功转型为架构师

详情：http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1

最新 最后发表 热门 精华

- 成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200% — 卷王1号 超级版主 前天 17:41
- 成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W — 卷王1号 超级版主 2022-9-21
- 成功案例：[878号卷王] 小伙8年经验，年薪60W — 卷王1号 超级版主 2022-8-13
- 年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W — 卷王1号 超级版主 2022-2-11
- 成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬季逆涨30% — 卷王1号 超级版主 前天 17:43
- 成功案例：[250号卷王] 就业极寒时代，收offer 涨25% — 卷王1号 超级版主 前天 17:38
- 成功案例：[612号卷王] 就业极寒时代，从外包到自研 — 卷王1号 超级版主 前天 17:15
- 成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W — 卷王1号 超级版主 2022-9-21
- 成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100% — 卷王1号 超级版主 2022-9-21
- 成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K — 卷王1号 超级版主 2022-9-21
- 成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30% — 卷王1号 超级版主 2022-9-21
- 成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺 — 卷王1号 超级版主 2022-5-30
- 成功案例：[1002号卷王] 5月份"被毕业"，改简历后，斩获顶级央企Offer，涨薪7000+ — 卷王1号 超级版主 2022-7-5
- 成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120% — 卷王1号 超级版主 2022-8-13
- 成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭 — 卷王1号 超级版主 2022-7-6
- 成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺 — 卷王1号 超级版主 2022-5-30
- 成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺 — 卷王1号 超级版主 2022-5-21
- 成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺 — 卷王1号 超级版主 2022-5-21
- 成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺 — 卷王1号 超级版主 2022-5-12
- 成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺 — 卷王1号 超级版主 2022-5-13
- 成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60% — 卷王1号 超级版主 2022-5-11
- 成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家 — 卷王1号 超级版主 2022-4-21

最新 最后发表 热门 精华

- 成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W — 卷王1号 超级版主 2022-4-16
- 成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万 — 卷王1号 超级版主 2022-4-14
- 成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer — 卷王1号 超级版主 2022-4-12
- 成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer — 卷王1号 超级版主 2022-4-7
- 成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer — 卷王1号 超级版主 2022-3-25
- 成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer — 卷王1号 超级版主 2022-3-24
- 成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer — 卷王1号 超级版主 2022-3-21
- 成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+ — 卷王1号 超级版主 2022-3-20
- 成功案例：热烈祝贺1群小卷王offer拿到手软，甚至拒了阿里offer — 卷王1号 超级版主 2022-3-16
- 简历案例：简历一改，腾讯的邀请就来了！热烈祝贺，小伙收到一大堆面试邀请 — 卷王1号 超级版主 2022-3-10
- 成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面 — 卷王1号 超级版主 2022-3-10
- 成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer — 卷王1号 超级版主 2022-3-10
- 成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer — 尼恩 超级版主 2022-3-5
- 成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到宇节，年薪45W — 卷王1号 超级版主 2022-2-27
- 成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+ — 卷王1号 超级版主 2022-2-27
- 成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端 — 卷王1号 超级版主 2022-2-27
- 成功案例：狠卷1年技术，喜收 腾讯、阿里、微软三大Offer，最高年薪56W — 卷王1号 超级版主 2022-2-27
- 成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer，年薪33W — 卷王1号 超级版主 2022-2-27

- 成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer，年薪35W — 卷王1号 超级版主 2022-2-27
- 成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[569号卷王] 双非应届生卷王，喜收字节跳动实习offer — 卷王1号 超级版主 2022-2-25
- 成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元！ — 卷王1号 超级版主 2022-2-25
- 成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K — 卷王1号 超级版主 2022-2-10

- 成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习 — 卷王1号 超级版主 2022-2-10
- 成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪 — 卷王1号 超级版主 2022-2-10
- 成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer — 卷王1号 超级版主 2022-2-10
- 成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer — 卷王1号 超级版主 2022-2-10
- [612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈 — 卷王1号 超级版主 2022-2-10

简历优化后的成功涨薪案例（VIP 含免费简历优化）

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- **从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨**的指导能力。
- **尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。**

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：https://www.yuque.com/crazymakercircle/gkkw8s/khigna
码云：https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md