**Mybatis知识总结**

### 1.基础简介

#### 1.1 什么是Mybatis

一个针对sql,存储过程和映射的框架,封装好了jdbc的各种繁琐设置.使用简单的xml和注解,将接口和java的pojos映射成数据库中的记录.

#### 1.2 Mybatis如何安装

mybatis比较小,也没啥依赖,用的时候将所需版本的mybatis-x.x.x.jar包置于classpath，或者项目用Maven构建的时候，将其置于pom.xml中

#### 1.3 Mybatis功能架构

一般分为三层



* API接口层：提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。

主要使用SqlSession接口，通过SqlSession和Mapper接口，可以同时Mybatis调用一条sql命令以及sql命令关联参数

具体使用方式

public void start() throws IOException{

InputStream is=Resources.getResourceAsStream("myBatis-config.xml");

SqlSessionFactory factory=new SqlSessionFactoryBuilder().build(is);

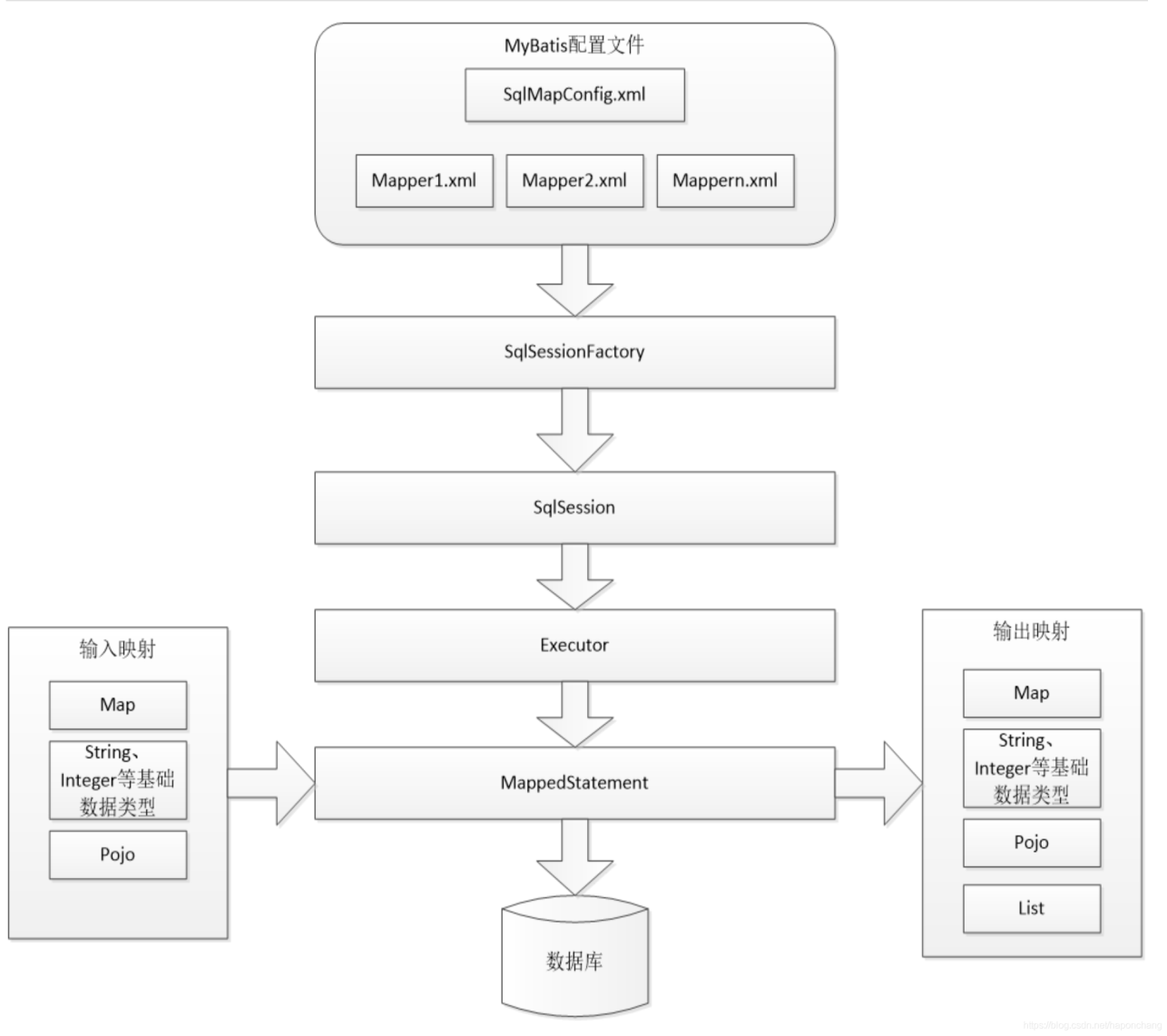
session=factory.openSession();

session.insert("InsertDept\_Sql",dept)

}

* 数据处理层：负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。
* 基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

#### 1.4Mybatis运行流程



说明：

1.mybatis配置文件

SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis  
的运行环境等信息。  
 Mapper.xml，此文件作为mybatis的sql映射文件，文件中配置了操作数据  
库的sql语句。此文件需要在SqlMapConfig.xml中加载。

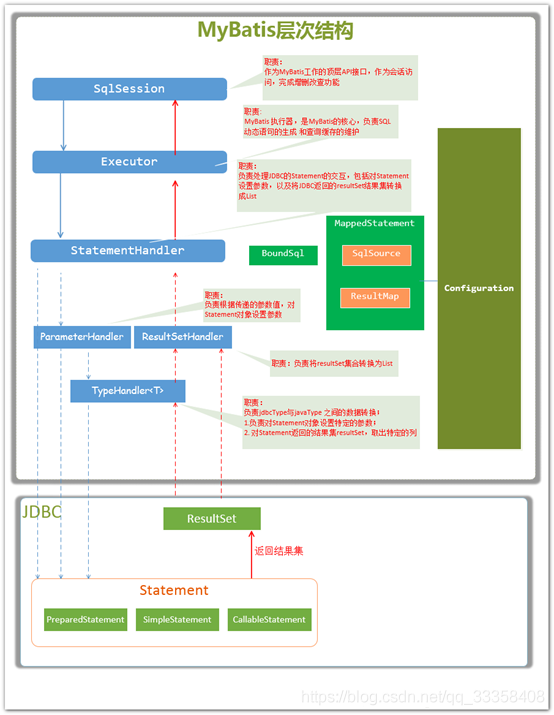
2.SqlSessionFactory  
通过mybatis环境等配置信息构造SqlSessionFactory，即会话工厂。

3.sqlSession  
通过会话工厂创建sqlSession即会话，程序员通过sqlsession会话接口对数据库  
进行增删改查操作。

 4.Executor执行器  
mybatis底层自定义了Executor执行器接口来具体操作数据库，Executor接口有  
两个实现，一个是基本执行器（默认）、一个是缓存执行器，sqlsession底层是  
通过executor接口操作数据库的。

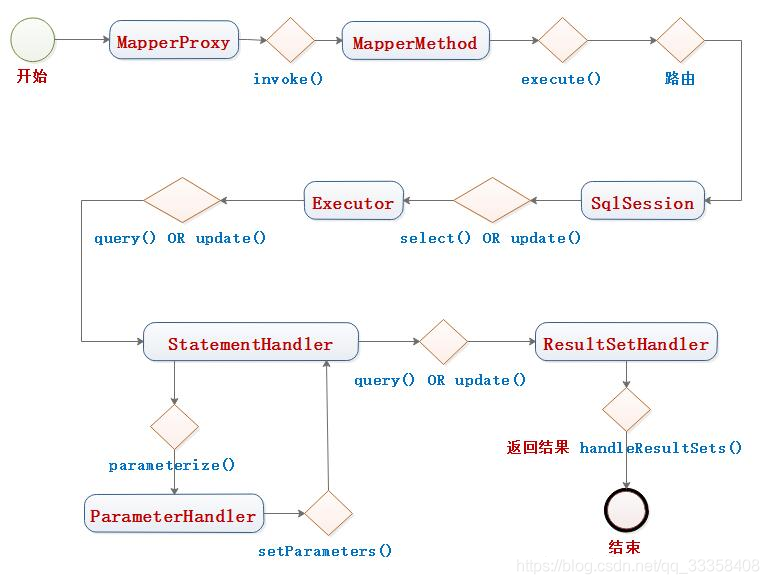
 5. Mapped Statement  
 它也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个select\insert\update\delete标签对应一个Mapped Statement对象， select\insert\update\delete 标签的id即是Mappedstatement的id。  
 Mapped Statement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中，输入参数映射就是jdbc编程中对preparedStatement设置参数。  
 Mapped Statement对sql执行输出结果进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql后将输出结果映射至java对象中，输出结果映射过程相当于jdbc编程中对结果的解析处理过程。

调用流程图



SqlSession  
 接收开发人员提供Statement Id 和参数.并返回操作结果  
Executor  
 MyBatis执行器，是MyBatis 调度的核心，负责SQL语句的生成和查询缓存的维护  
StatementHandler  
 封装了JDBC Statement操作，负责对JDBC statement 的操作，如设置参数、将Statement结果集转换成List集合。  
ParameterHandler  
 负责对用户传递的参数转换成JDBC Statement 所需要的参数  
ResultSetHandler  
 负责将JDBC返回的ResultSet结果集对象转换成List类型的集合  
TypeHandler  
 负责java数据类型和jdbc数据类型之间的映射和转换  
MappedStatement  
 维护了一条<select|update|delete|insert>节点的封装  
SqlSource  
 负责根据用户传递的parameterObject，动态地生成SQL语句，将信息封装到BoundSql对象中，并返回BoundSql表示动态生成的SQL语句以及相应的参数信息  
Configuration  
 MyBatis所有的配置信息都维持在Configuration对象之中。

mapper代理执行原理



### 2.入门

#### 2.1 从xml中构建SqlSessionFactory

每个基于MyBatis的应用都是以一个SqlSessionFactory 的实例为核心的。

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

### 3.问题

#### 3.1 常见问题

##### 3.1.1 大于号、小于号在sql中的转换

sql写在xml中，特殊字符会引起xml格式错误。有两种方法解决

（1）转义字符

select \* from product where price &gt;= #{minprice} and price &lt;=#{maxPrice}

(2)标记CDATA

<![CDATA[select \* from product where price>gt;= #{minprice} and price <=#{maxPrice}]]>

转义字符表

|  |  |
| --- | --- |
| **&lt;** | **<** |
| **&gt;** | **>** |
| **&amp;** | **&** |
| **&apos;** | **’** |
| **&quot;** | **“** |

##### 3.1.2 MyBatis中的resultType和resultMap

简单说，resultType用于返回值只有一个字段，resultMap用于返回值是一个map。

使用resultType

<select id="count" resultType="java.lang.Integer">    
         SELECT count(\*) FROM USER    
</select>

使用resultMap

<resultMap type="com.liulanghan.Blog" id="BlogResult">      
     <id column="id" property="id"/>      
     <result column="title" property="title"/>      
     <result column="content" property="content"/>      
     <result column="owner" property="owner"/>      
 </resultMap>      
 <select id="selectBlog" parameterType="int" resultMap="BlogResult">      
       select \* from t\_blog where id = #{id}      
 </select>

##### 3.1.3 参数

Mapper中需要用@Param("queryDate")定义参数名称，sql 中用#{queryDate}使用参数，字符串也不需要引号。

参数判断和if的用法：

<if test="queryDate != null">  
     and queryDate >= #{queryDate}  
 </if>

when otherwise 就是 if else

<choose>  
     <when test="isDelete != null and isDelete == 0">  
           isDelete=0  
       </when>  
       <otherwise>  
           isDelete=1  
       </otherwise>  
</choose>

如果要判断的字符串，则需要加引号

<when test="gender != null and gender == 'MALE'">  
    gender='MALE'  
</when>

##### 3.1.4 传入参数为0查询条件失效

场景：需要做一个对账单查询，可以按金额范围进行查询，页面参数写完之后进行条件，输入0测试了无数次均失效。

原因解析: 当页面参数为 0，传入到 mybatis 的 xml 中后，如果不是字符串，需指定数据类型，否则会被误认为 null

<if test="data.tatalAmount != null and data.totalAmount !='' ">

and total\_Amount=#{data.totalAmount}

</if>

这种情况如果 totalAmount 为 0 时将被误认为是 null，里面的条件不会执行。

正确的姿势如下

1.添加0判断

<if test="data.tatalAmount != null and data.totalAmount !='' or tatalAmount==0 ">

and total\_Amount=#{data.totalAmount}

</if>

2规定传入参数的类型

<if test="data.tatalAmount != null and data.totalAmount !='' ">

and total\_Amount=#{data.totalAmount,jdbc.Type=DECIMAL}

</if>

#### 3.2 Mybatis和ORM区别

mybatis是半orm，因为sql需要自己写。标准的orm（hibernate）将java对象与数据库关联起来。而mybatis是把java方法与sql关联起来。

允许用户利用数据库的存储过程，视图和各种复杂的查询以及某些专有特性。

自己写的sql灵活性很高，但是平台移植性不高。

#### 3.3 Mybatis实现分页功能

三种方法

原始方法：使用limit

select \* from table limit 5;

拦截StatemenHandler，本质还是最后生成limit语句

使用PageHelper插件，最主流的方法。

#### 3.3 Mybatis四种现分页方式

\*\*\*

#### 3.4 Mybatis中#{} 和${}区别

#{} 是预编译处理，传进来的数据会加个"";

${} 就是字符串替换。直接替换掉占位符。一般用于传入数据库对象，比如表名。

为了泛指sql注入，能用#{} 的不要用${}

#### 3.5 Mybatis接口绑定的方式

接口绑定有两种方式

1使用注解，在接口的方法上面添加@Select、@Update

2.通过映射文件xml方式进行绑定，指定xml文件中的namespace对应的接口

当sql比较简单的时候，使用注解绑定，当sql比较复杂的话，使用xml方式绑定。

#### 3.5 Mybatis动态sql语句

1.if

 <select id="select" resultType="Blog">

 SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

 <if test="title != null">

 AND title like #{title}

 </if>

 </select>

2.wehere

 <select id="select" resultType="Blog">

 SELECT \* FROM BLOG

 <where>

 <if test="title != null">

 AND title like #{title}

 </if>

 <where>

 </select>

3.choose（when、otherwise）

 <select id="findActiveBlogLike"resultType="Blog">

 SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

 <choose>

 <when test="title != null">

 AND title like #{title}

 </when>

 <when test="author != null and author.name != null">

 AND author\_name like #{author.name}

 </when>

 <otherwise>

 AND featured = 1

 </otherwise>

 </choose>

 </select>

4.tirm : prefix：前缀prefixoverride：去掉第一个and或者是or

select \* from test

<trim prefix="WHERE" prefixoverride="AND丨OR">

<if test="a!=null and a!=' '">AND a=#{a}<if>

<if test="b!=null and b!=' '">AND a=#{a}<if>

</trim>

5.set

set 元素主要是用在更新操作的时候，如果包含的语句是以逗号结束的话将会把该逗号忽略，如果set包含的内容为空的话则会出错。

 <update id="dynamicSetTest" parameterType="Blog">

 update t\_blog

 <set>

 <if test="title != null">

 title = #{title},

 </if>

 <if test="content != null">

 content = #{content},

 </if>

 <if test="owner != null">

 owner = #{owner}

 </if>

 </set>

 where id = #{id}

 </update>

6. foreach

foreach主要用在构建in条件中

 <select id="dynamicForeachTest" resultType="Blog">

 select \* from t\_blog where id in

 <foreach collection="list" index="index" item="item" open="("separator="," close=")">

 #{item}

 </foreach>

 </select>

open separator close

相当于是in （？，？，？）

 如果是个map怎么办

 <select id="dynamicForeach3Test" resultType="Blog">

 select \* from t\_blog where title like "%"#{title}"%" and id in

 <foreach collection="ids" index="index" item="item" open="(" separator="," close=")">

 #{item}

 </foreach>

 </select>

#### 3.6 Mybatis 中Like有几种方法

1. LIke '%${name}%'

2.   LIKE "%"#{name}"%"

3. 字符串拼接

AND name LIKE CONCAT(CONCAT('%',#{name},'%'))

4.bind标签

 <select id="searchStudents" resultType="com.example.entity.StudentEntity"

 parameterType="com.example.entity.StudentEntity">

 <bind name="pattern1" value="'%' + \_parameter.name + '%'" />

 <bind name="pattern2" value="'%' + \_parameter.address + '%'" />

 SELECT \* FROM test\_student

 <where>

 <if test="age != null and age != '' and compare != null and compare != ''">

 age

 ${compare}

 #{age}

 </if>

 <if test="name != null and name != ''">

 AND name LIKE #{pattern1}

 </if>

 <if test="address != null and address != ''">

 AND address LIKE #{pattern2}

 </if>

 </where>

 ORDER BY id

 </select>

5.java代码里写

param.setUsername("%CD%"); 在 java 代码中传参的时候直接写上

<if test="username!=null"> AND username LIKE #{username}</if>

然后 mapper 里面直接写 #{} 就可以了

#### 3.7 通常一个mapper.xml对应一个DAO接口，DAO是否可以重载

不能重载，Mapper接口不能重载，因为使用 全限名+方法名的保存和寻找策略。

Mapper接口的工作原理是JDK动态代理。Mybatis 运行时会使用 JDK动态代理为 Mapper 接口生成代理对象 proxy，代理对象会拦截接口方法，转而执行 MapperStatement 所代表的 sql，然后将 sql 执行结果返回。

#### 3.8 MyBatis映射文件中A标签引用B标签，B可以在A后面定义吗？

虽然 Mybatis 解析 [Xml 映射文件](https://www.w3cschool.cn/mybatis/f4uw1ilx.html)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：

Mybatis 解析 A 标签时，发现引用了 B 标签，未解析到 B 标签，此时会把 A 标签标记为未解析状态；

继续解析下面内容，把剩下解析完之后，再解析标记为未解析的标签；

此时已解析到 B 标签，此时再解析A标签时，B标签已经存在，A 标签也就顺利解析完成。

#### 3.9 MyBatis映射文件中id是否可以重复？

**可以重复，但是需要映射文件的namespace不同**

不同的 [Xml 映射文件](https://www.w3cschool.cn/mybatis/f4uw1ilx.html)，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。

原因就是 namespace+id 是作为 Map<String, MapperStatement>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。

#### 3.10 MyBatis是否可以映射到枚举类

Mybatis 可以映射枚举类。

不单可以映射枚举类，Mybatis 可以映射任何对象到表的一列上。映射方式为自定义一个 TypeHandler ，实现 TypeHandler 的 ​setParameter()​ 和 ​getResult()​ 接口方法。

TypeHandler 有两个作用，一是完成从 javaType 至 jdbcType 的转换，二是完成 jdbcType 至 javaType 的转换，体现为 ​setParameter() ​和 ​getResult()​ 两个方法，分别代表设置 sql 问号占位符参数和获取列查询结果。

#### 3.11 MyBatis如何获取自动生成的主角id

MySQL：Mapper 文件 insert 语句设置

useGeneratedKeys="true" keyProperty="id"

Oracle：Mapper 文件 insert 语句增加

 <selectKey keyProperty="id" order="BEFORE" resultType="Integer">

     select xxx\_SEQ.nextval from dual

 </selectKey>

#### 3.12 MyBatis如何传递多个参数

方法1：使用map接口传递

方法2：使用注解传递

方法3：通过javaBean传递

方法4：混合使用

总结:

使用 map 传递参数导致了业务可读性的丧失，导致后续扩展和维护的困难，在实际的应用中要果断废弃这种方式。

　　使用 @Param 注解传递多个参数，受到参数个数（n）的影响。当 n≤5 时，这是最佳的传参方式，它比用 Java Bean 更好，因为它更加直观；当 n＞5 时，多个参数将给调用带来困难，此时不推荐使用它。

　　当参数个数多于5个时，建议使用 Java Bean 方式。

　　对于使用混合参数的，要明确参数的合理性。

#### 3.13 MyBatis缓存机制

缓存机制减轻数据库压力，提高数据库性能

mybatis的缓存分为两级：一级缓存、二级缓存

一级缓存：

一级缓存为 ​sqlsesson​ 缓存，缓存的数据只在 SqlSession 内有效。在操作数据库的时候需要先创建 SqlSession 会话对象，在对象中有一个 HashMap 用于存储缓存数据，此 HashMap 是当前会话对象私有的，别的 SqlSession 会话对象无法访问。

具体流程：

第一次执行 select 完毕会将查到的数据写入 SqlSession 内的 HashMap 中缓存起来

第二次执行 select 会从缓存中查数据，如果 select 同传参数一样，那么就能从缓存中返回数据，不用去数据库了，从而提高了效率

注意：

1、如果 SqlSession 执行了 DML 操作（insert、update、delete），并 commit 了，那么 mybatis 就会清空当前 SqlSession 缓存中的所有缓存数据，这样可以保证缓存中的存的数据永远和数据库中一致，避免出现差异

2、当一个 SqlSession 结束后那么他里面的一级缓存也就不存在了， mybatis 默认是开启一级缓存，不需要配置

3、 mybatis 的缓存是基于 [namespace:sql语句:参数] 来进行缓存的，意思就是， SqlSession 的 HashMap 存储缓存数据时，是使用 [namespace:sql:参数] 作为 key ，查询返回的语句作为 value 保存的

二级缓存：

二级缓存是​ mapper​ 级别的缓存，也就是同一个 namespace 的 mapper.xml ，当多个 SqlSession 使用同一个 Mapper 操作数据库的时候，得到的数据会缓存在同一个二级缓存区域

二级缓存默认是没有开启的。需要在 setting 全局参数中配置开启二级缓存

开启二级缓存步骤：

1、​conf.xml​ 配置全局变量开启二级缓存

<settings>  
 <setting name="cacheEnabled" value="true"/>默认是false：关闭二级缓存  
<settings>

2、在​ userMapper.xml ​中配置

<cache eviction="LRU" flushInterval="60000" size="512" readOnly="true"/>当前mapper下所有语句开启二级缓存

这里配置了一个 LRU 缓存，并每隔60秒刷新，最大存储512个对象，而返回的对象是只读的

若想禁用当前​select​语句的二级缓存，添加 ​useCache="false"​修改如下：

<select id="getCountByName" parameterType="java.util.Map" resultType="INTEGER" statementType="CALLABLE" useCache="false">

具体流程：

1.当一个​ sqlseesion ​执行了一次​ select​ 后，在关闭此​ session​ 的时候，会将查询结果缓存到二级缓存

2.当另一个​ sqlsession ​执行​ select​ 时，首先会在他自己的一级缓存中找，如果没找到，就回去二级缓存中找，找到了就返回，就不用去数据库了，从而减少了数据库压力提高了性能

注意:

1、如果 ​SqlSession​ 执行了 DML 操作​（insert、update、delete）​，并 ​commit​ 了，那么 ​mybatis​ 就会清空当前​ mapper​ 缓存中的所有缓存数据，这样可以保证缓存中的存的数据永远和数据库中一致，避免出现差异

2、​ mybatis​ 的缓存是基于​ [namespace:sql语句:参数] ​来进行缓存的，意思就是，​SqlSession​ 的 ​HashMap​ 存储缓存数据时，是使用 ​[namespace:sql:参数] ​作为 ​key​ ，查询返回的语句作为 ​value​ 保存的。

#### 3.14 MyBatis时间timestamp做条件进行查询

首先要将条件 转换为 时间戳

 long startTime = TimeUtil.parseTimestamp(start);

 long endTime = TimeUtil.parseTimestamp(end);



 /\*对应工具类\*/

 public static long parseTimestamp(String datetime){

 try{

 SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

 Date date = dateformat.parse(datetime);

 return date.getTime()/1000;

 }catch(Exception e){

 e.printStackTrace();

 }



 return 0;

 }

然后Mapper.xml中  使用BETWEEN and  和 to\_timestamp

 <if test="startDate !=null and startDate !='' and endDate !=null and endDate !=''">

 AND tdnm.create\_time BETWEEN to\_timestamp(#{startDate}) AND to\_timestamp(#{endDate})

 </if>

### 4.狂神说

#### 4.1 常见问题