[今日课程大纲]

MyBatis 接口绑定方案及多参数传递

动态 SQL

缓存

ThreadLocal 讲解

使用 Filter 模拟 OpenSessionInView 实现编程式事务

[知识点详解]

一.MyBatis 接口绑定方案及多参数传递

1.作用:实现创建一个接口后把 mapper.xml 由 mybatis 生成接口的实现类,通过调用接口对象就可以获取 mapper.xml 中编写的 sql.

2.后面 mybatis 和 spring 整合时使用的是这个方案.

3.实现步骤:

3.1 创建一个接口

3.1.1 接口包名和接口名与 mapper.xml 中<mapper>namespace 相同

3.1.2 接口中方法名和 mapper.xml 标签的 id 属性相同

3.2 在 mybatis.xml 中使用<package>进行扫描接口和 mapper.xml

4.代码实现步骤:

4.1 在 mybatis.xml 中<mappers>下使用<package>

```
<mappers>
```

```
    <package name="com.bjsxt.mapper"/>
  </mappers>
```

4.2 在 com.bjsxt.mapper 下新建接口

```
public interface LogMapper {
  List<Log> selAll();
}
```

4.3 在 com.bjsxt.mapper 新建一个 LogMapper.xml

4.3.1 namespace 必须和接口全限定路径(包名+类名)一致

4.3.2 id 值必须和接口中方法名相同

4.3.3 如果接口中方法为多个参数,可以省略 parameterType

```
<mapper namespace="com.bjsxt.mapper.LogMapper">
  <select id="selAll" resultType="log">
    select * from log
  </select>
</mapper>
```

5.多参数实现办法

5.1 在接口中声明方法

```
List<Log> selByAccInAccout(String accin,String accout);
```

5.2 在 mapper.xml 中添加

5.2.1 #{}中使用 0,1,2 或 param1,param2

```
  <!-- 当多参数时,不需要写 parameterType -->
```

```
<select id="selByAccInAccout" resultType="log" >
    select * from log where accin=#{0}  and accout=#{1}
</select>
```

6,可以使用注解方式

6.1 在接口中声明方法

```
/**
 * mybatis 把参数转换为 map 了,其中@Param("key") 参数内容就是 map 的 value
 * @param accin123
 * @param accout3454235
 * @return
 */
List<Log> selByAccInAccout(@Param("accin") String accin123,@Param("accout") String accout3454235);
```

6.2 在 mapper.xml 中添加

6.2.1 #{} 里面写@Param(“内容”)参数中内容

```
<!-- 当多参数时,不需要写 parameterType -->
<select id="selByAccInAccout" resultType="log" >
    select * from log where accin=#{accin}  and accout=#{accout}
</select>
```

二.动态 SQL

1. 根据不同的条件需要执行不同的 SQL 命令.称为动态 SQL
2. MyBatis 中动态 SQL 在 mapper.xml 中添加逻辑判断等.
3. If 使用

```
<select id="selByAccinAccout" resultType="log">
      select * from log where 1=1
      <!-- OGNL 表达式,直接写 key 或对象的属性.不需要添加任何特字符号 -->
      <if test="accin!=null and accin!=''">
         and accin=#{accin}
      </if>
      <if test="accout!=null and accout!=''">
         and accout=#{accout}
      </if>
   </select>
```

4. <where>

   4.1 当编写 where 标签时,如果内容中第一个是 and 去掉第一个 and

   4.2 如果<where>中有内容会生成 where 关键字,如果没有内容不生成 where 关键

   4.3 使用示例

4.3.1 比直接使用<if>少写 where 1=1

```
<select id="selByAccinAccout" resultType="log">
   select * from log
   <where>
      <if test="accin!=null and accin!=''">
         and accin=#{accin}
      </if>
      <if test="accout!=null and accout!=''">
         and accout=#{accout}
      </if>
   </where>
</select>
```

5. <choose> <when> <otherwise>

5.1 只有有一个成立,其他都不执行.

5.2 代码示例

5.2.1 如果 accin 和 accout 都不是 null 或不是""生成的 sql 中只有 where accin=?

```
<select id="selByAccinAccout" resultType="log">
   select * from log
   <where>
      <choose>
         <when test="accin!=null and accin!=''">
```

```
            and accin=#{accin}
        </when>
        <when test="accout!=null and accout!=''">
            and accout=#{accout}
        </when>
    </choose>
</where>
</select>
```

6. <set>用在修改 SQL 中 set 从句

6.1 作用:去掉最后一个逗号

6.2 作用:如果<set>里面有内容生成 set 关键字,没有就不生成

6.3 示例

6.3.1 id=#{id} 目的防止<set>中没有内容,mybatis 不生成 set 关键字,如果修改中没有 set 从句 SQL 语法错误.

```
<update id="upd" parameterType="log" >
    update log
    <set>
        id=#{id},
        <if test="accIn!=null and accIn!=''">
            accin=#{accIn},
        </if>
        <if test="accOut!=null and accOut!=''">
```

```
            accout=#{accOut},
        </if>
    </set>
    where id=#{id}
</update>
```

7. Trim

7.1 prefix 在前面添加内容

7.2 prefixOverrides 去掉前面内容

7.3 suffix 在后面添加内容

7.4 suffixOverrieds 去掉后面内容

7.5 执行顺序去掉内容后添加内容

7.6 代码示例

```
<update id="upd" parameterType="log">
    update log
    <trim prefix="set" suffixOverrides=",">
    a=a,
    </trim>
    where id=100
</update>
```

8. <bind>

8.1 作用:给参数重新赋值

8.2 场景:

8.2.1 模糊查询

8.2.2 在原内容前或后添加内容

8.3 示例

```
<select id="selByLog" parameterType="log"
resultType="log">
    <bind name="accin" value="'%'+accin+'%'"/>
    #{money}
</select>
```

9. <foreach>标签

9.1 循环参数内容,还具备在内容的前后添加内容,还具备添加分隔符功能.

9.2 适用场景:**in 查询中**.批量新增中(mybatis 中 foreach 效率比较低)

9.2.1 如果希望批量新增,SQL 命令

```
insert into log VALUES
(default,1,2,3),(default,2,3,4),(default,3,4,5)
```

9.2.2 openSession()必须指定

9.2.2.1 底层 JDBC 的 PreparedStatement.addBatch();

```
factory.openSession(ExecutorType.BATCH);
```

9.3 示例

9.3.1 collectino="" 要遍历的集合

9.3.2 item 迭代变量, #{迭代变量名}获取内容

9.3.3 open 循环后左侧添加的内容

9.3.4 close 循环后右侧添加的内容

9.3.5 separator 每次循环时,元素之间的分隔符

```
<select id="selIn" parameterType="list"
resultType="log">
    select * from log where id in
    <foreach collection="list" item="abc" open="("
close=")" separator=",">
        #{abc}
    </foreach>
</select>
```

10. <sql> 和<include>

10.1 某些 SQL 片段如果希望复用,可以使用<sql>定义这个片段

```
<sql id="mysql">
    id,accin,accout,money
</sql>
```

10.2 在<select>或<delete>或<update>或<insert>中使用<include>引用

```
<select id="">
    select <include refid="mysql"></include>
    from log
```

```
</select>
```

三 ThreadLocal

1. 线程容器,给线程绑定一个 Object 内容,后只要线程不变,可以随时取出.

   1.1 改变线程,无法取出内容.

2. 语法示例

```
final ThreadLocal<String> threadLocal = new ThreadLocal<>();
threadLocal.set("测试");
new Thread(){
    public void run() {
        String result = threadLocal.get();
        System.out.println("结果:"+result);
    };
}.start();
```

四.缓存

1. 应用程序和数据库交互的过程是一个相对比较耗时的过程

2. 缓存存在的意义:让应用程序减少对数据库的访问,提升程序运行

效率

3. MyBatis 中默认 SqlSession 缓存开启

3.1 同一个 SqlSession 对象调用同一个<select>时,只有第一次访问数据库,第一次之后把查询结果缓存到 SqlSession 缓存区(内存)中

3.2 缓存的是 statement 对象.(简单记忆必须是用一个<select>)

3.2.1 在 myabtis 时一个<select>对应一个 statement 对象

3.3 有效范围必须是同一个 SqlSession 对象

4. 缓存流程

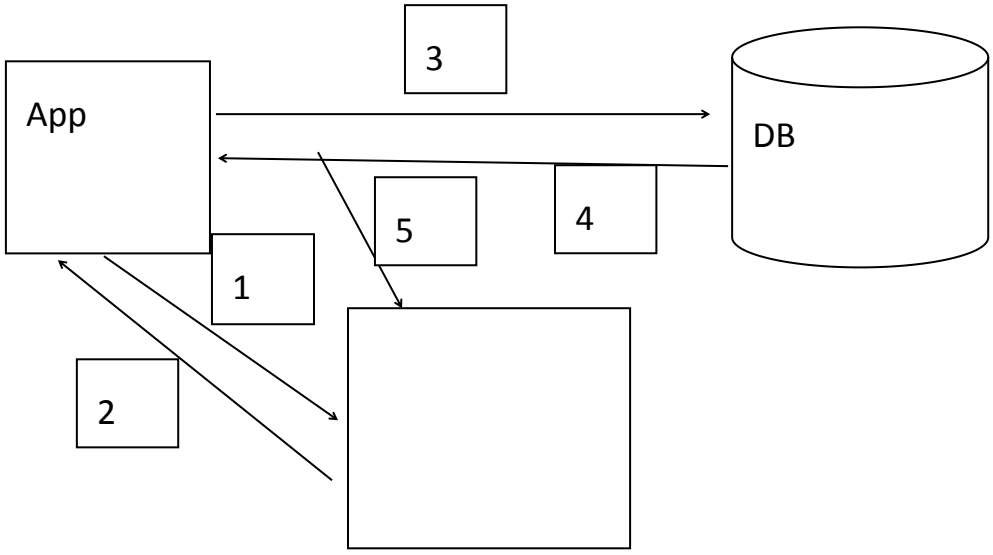
4.1 步骤一: 先去缓存区中找是否存在 statement

4.2 步骤二:返回结果

4.3 步骤三:如果没有缓存 statement 对象,去数据库获取数据

4.4 步骤四:数据库返回查询结果

4.5 步骤五:把查询结果放到对应的缓存区中



5. SqlSessionFactory 缓存

5.1 又叫:二级缓存

5.2 有效范围:同一个 factory 内哪个 SqlSession 都可以获取

5.3 什么时候使用二级缓存:

5.3.1 当数据频繁被使用,很少被修改

5.4 使用二级缓存步骤

5.4.1 在 mapper.xml 中添加

5.4.2 如果不写 readOnly=”true”需要把实体类序列化

```
<cache readOnly="true"></cache>
```

5.5 当 SqlSession 对象 close()时或 commit()时会把 SqlSession 缓存的数据刷(flush)到 SqlSessionFactory 缓存区中