mybatis 总结

环境搭建步骤 (xml 方式)

1.1)导入jar

```xml
<dependencies>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.4.5</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>

        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.12</version>
        </dependency>

        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
        </dependency>
    </dependencies>
```

1.2 编写核心配置文件

```
select,
udpate
delete
insert,selectKey
resultmap
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

```
        <!DOCTYPE configuration
                PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
                "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
<!-- 引入外部配置文件-->
<properties resource="jdbcConfig.properties"></properties>

  <settings>
        <!--开启Mybatis支持延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
        <setting name="aggressiveLazyLoading" value="false"></setting>
      <!-- 开启二级缓存-->
        <setting name="cacheEnabled" value="true"/>
    </settings>


<!--配置别名-->
<typeAliases>
    <package name="com.itheima.domain"></package>
</typeAliases>
<!-- 配置环境-->
<environments default="mysql">
    <environment id="mysql">
        <transactionManager type="JDBC"></transactionManager>
        <dataSource type="POOLED">
            <property name="driver" value="${jdbc.driver}"></property>
            <property name="url" value="${jdbc.url}"></property>
            <property name="username" value="${jdbc.username}"></property>
            <property name="password" value="${jdbc.password}"></property>
        </dataSource>
    </environment>
</environments>
<!-- 指定带有注解的dao接口所在位置 -->
<mappers>
    <!-- 指定配置文件所在的位置-->
    <package name="com.itheima.dao"></package>
</mappers>
</configuration>
```

1.3 创建dao 接口,domain 类

```

```

1.4 编写 dao.xml 文件

```
IRoleDao.xml (多对多为例)
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.dao.IRoleDao">
```

```
    <!--开启二级缓存-->
    <cache/>
    <resultMap id="roleMap" type="com.itheima.domain.Role">
        <id property="roleId" column="rid"/>
        <result property="roleName" column="role_name"/>
        <result property="roleDesc" column="role_Desc"/>
        <collection property="users" ofType="com.itheima.domain.User"
javaType="java.util.List">
            <id property="userId" column="id"></id>
            <result property="userName" column="username"/>
            <result property="userSex" column="sex"/>
            <result property="userAddress" column="address"/>
            <result property="userBirthday" column="birthday"/>
        </collection>
    </resultMap>

    <select id="findAll" resultMap="roleMap" >
      SELECT u.*, r.`ID` rid ,r.`ROLE_DESC`,r.`ROLE_NAME` FROM role r LEFT JOIN
    user_role ur ON r.`ID`=ur.`RID`
    LEFT JOIN  USER u ON ur.`UID`=u.`id`
    </select>

</mapper>
```

sql 常用的标签

```
select,
udpate
delete
insert,selectKey
resultmap()
```

```
<select id="findById" parameterType="INT" resultMap="userMap">

        select * from user where id = #{uid}

        <where>
            <if test="uid >0">
                <foreach
                         collection="user.vo"
                         open="and id in("
                         separator=","
                         close=")"
                         item="item">
                     #{item}
                </foreach>
            </if>

        </where>
</select>


<insert id="saveUser" parameterType="user">
        <!-- 配置插入操作后，获取插入数据的id -->
        <selectKey keyProperty="userId" keyColumn="id" resultType="int" order="AFTER">
            select last_insert_id();
        </selectKey>
        insert into user(username,address,sex,birthday)values(#{userName},#
{userAddress},#{userSex},#{userBirthday});
    </insert>

 <sql id="id">
    select * from user
  </sql>


```

一对一01

```
注意
    1) sql 返回字段名称尽量不要重复
    2) sql 字段和java 实体类映射关系
<resultMap id="accountMap" type="com.itheima.domain.Account">
        <id property="id" column="aid"></id>
        <result property="uid" column="uid"></result>
        <result property="money" column="money"></result>
        <association property="user" javaType="com.itheima.domain.User">
            <id property="id" column="uid" ></id>
            <result property="address" column="address"></result>
            <result property="sex" column="sex"></result>
            <result property="username" column="username"></result>
            <result property="birthday" column="birthday"></result>
        </association>
</resultMap>
<select id="findAll" resultMap="accountMap">
       SELECT a.*,b.uid,b.id aid,b.MONEY FROM USER a, account b WHERE a.id=b.uid
</select>
```

一对一02 (可开启延迟加载)

```
<!-- 定义封装account和user的resultMap -->
    <resultMap id="accountUserMap" type="account">
        <id property="id" column="id"></id>
        <result property="uid" column="uid"></result>
        <result property="money" column="money"></result>
        <!-- 一对一的关系映射：配置封装user的内容
        select属性指定的内容：查询用户的唯一标识：
        column属性指定的内容：用户根据id查询时，所需要的参数的值
        -->
        <association property="user" column="uid" javaType="user"
select="com.itheima.dao.IUserDao.findById">
        </association>
    </resultMap>
    <!-- 查询所有 -->
    <select id="findAll" resultMap="accountUserMap">
        select * from account
    </select>
```

```
两种一对一的差别
    1) sql 有区别: 延迟加载的sql 语句只负责当前对象的数据查询
    2) 配置有区别，
```

一对多01

```
<resultMap id="userMap" type="com.itheima.domain.User">
        <id property="id" column="id"/>
        <result property="username" column="username"/>
        <result property="sex" column="sex"/>
        <result property="address" column="address"/>
        <result property="birthday" column="birthday"/>

        <collection property="accounts"   ofType="com.itheima.domain.Account" >
            <id property="id" column="aid"></id>
            <result property="uid" column="uid"></result>
            <result property="money" column="money"></result>
        </collection>
    </resultMap>

    <select id="findAll" resultMap="userMap" useCache="true">
      SELECT a.*,b.id aid,b.`UID`,b.`MONEY` FROM USER a LEFT JOIN account  b ON
a.id=b.uid
    </select>
```

一对多02(可开启延迟加载)

```
 <!-- 定义User的resultMap-->
    <resultMap id="userAccountMap" type="user">
        <id property="id" column="id"></id>
        <result property="username" column="username"></result>
        <result property="address" column="address"></result>
        <result property="sex" column="sex"></result>
        <result property="birthday" column="birthday"></result>
        <!-- 配置user对象中accounts集合的映射 -->
        <collection property="accounts" ofType="account"
select="com.itheima.dao.IAccountDao.findAccountByUid" column="id"></collection>
    </resultMap>

    <!-- 查询所有 -->
    <select id="findAll" resultMap="userAccountMap">
        select * from user
    </select>
```

```
<select id="findAccountByUid" resultType="account">
    select * from account where uid = #{uid}
</select>
```

注意

```
 <collection property="accounts" ofType="account"
select="com.itheima.dao.IAccountDao.findAccountByUid" column="id"/>
 此处的 column="id"  只的时sql 语句中返回字段的值,而不是实体类属性
```

多对多01

```
  <resultMap id="roleMap" type="com.itheima.domain.Role">
        <id property="roleId" column="rid"/>
        <result property="roleName" column="role_name"/>
        <result property="roleDesc" column="role_Desc"/>
        <collection property="users" ofType="com.itheima.domain.User"
javaType="java.util.List">
            <id property="userId" column="id"></id>
            <result property="userName" column="username"/>
            <result property="userSex" column="sex"/>
            <result property="userAddress" column="address"/>
            <result property="userBirthday" column="birthday"/>
        </collection>
    </resultMap>

    <select id="findAll" resultMap="roleMap" >
      SELECT u.*, r.`ID` rid ,r.`ROLE_DESC`,r.`ROLE_NAME` FROM role r LEFT JOIN
    user_role ur ON r.`ID`=ur.`RID`
    LEFT JOIN  USER u ON ur.`UID`=u.`id`
    </select>
```

多对多02 (可开启延迟加载)

```
 <resultMap id="roleMap" type="com.itheima.domain.Role">
        <id property="roleId" column="rid"/>
        <result property="roleName" column="role_name"/>
        <result property="roleDesc" column="role_Desc"/>
        <collection property="users" ofType="com.itheima.domain.User"
select="com.itheima.dao.IAccountDao.findAccountByUid" column="rid">
        </collection>
    </resultMap>

    <select id="findAll" resultMap="roleMap" >
      select *  from role
    </select>
```

```
<select id="findUserByRoleid" resultType="user">
  SELECT * FROM USER WHERE id IN (
        SELECT uid FROM user_role WHERE  rid=1
    )
</select>
```

延迟加载

```
 1)<settings>
        <!--开启Mybatis支持延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
        <setting name="aggressiveLazyLoading" value="false"></setting>
 </settings>
 2) dao.xml 中配置(详见上述说明)
```

缓存

一级缓存

sqlsession 中 默认开启

清除一级缓存的方式: sqlSession.clearCache();

增删改也会触发清除一级缓存

二级缓存

```
第一步：让Mybatis框架支持二级缓存（在SqlMapConfig.xml中配置）
<settings>
<setting name="cacheEnabled" value="true"/>
</settings>
第二步：让当前的映射文件支持二级缓存（在IUserDao.xml中配置）
<!--开启user支持二级缓存-->
<cache/>
第三步：让当前的操作支持二级缓存（在select标签中配置是否关闭,只配置上述两步时开启全部方法的缓存）
<!-- 根据id查询用户 -->
<select id="findById" parameterType="INT" resultType="user" useCache="true">
select * from user where id = #{uid}
</select>
```

```
二级缓存的注解开启方法
@CacheNamespace(blocking = true)
public interface IUserDao {}
```

注解方式配置

一对一

```
@Select("select * from account")
   @Results(id="accountMap",value = {
       @Result(id=true,column = "id",property = "id"),
       @Result(column = "uid",property = "uid"),
       @Result(column = "money",property = "money"),
       @Result(property = "user",column = "uid",
               one=@One(select="com.itheima.dao.IUserDao.findById",
                       fetchType= FetchType.EAGER)
               )
   })
   List<Account> findAll();
```

一对多

```
@Select("select * from user")
    @Results(id="userMap",value={
          @Result(id=true,column = "id",property = "userId"),
          @Result(column = "username",property = "userName"),
          @Result(column = "address",property = "userAddress"),
          @Result(column = "sex",property = "userSex"),
          @Result(column = "birthday",property = "userBirthday"),
          @Result(property = "accounts",column = "id",
                many = @Many(select = "com.itheima.dao.IAccountDao.findAccountByUid",
                           fetchType = FetchType.LAZY))
    })
    List<User> findAll();
```

```
一对多查询accounts
@Select("select * from account where uid = #{userId}")
    List<Account> findAccountByUid(Integer userId);
```

```
多对多查询roles
@Select("SELECT * FROM role WHERE id IN (SELECT rid FROM user_role WHERE uid=#{uid})")
    List<Role> findRolesByUid(Integer uid);
```