day02

第一天内容复习

1. mybatis 的配置文件
    - properties:加载路径下的properties文件，把该文件中的数据加载到Configuration对象中
    - Settings:mybatis运行日志，驼峰命名规则的设置，文件中的数据加载到Configuration对象中
    - typeAlias:别名映射，Mybatis默认有内嵌别名(String-string HashMap-map等)，通过指定package的包路径，(com.javasm不用指定详细的目录)，对包下的类做别名映射------>不同包下不允许出现同名的类。
    - environments: 数据库环境配置，文件中的数据加载到Configuration对象中
    - Mappers：映射文件的引入，文件中的数据加载到Configuration对象中的mappedStatements成员变量中，Map<String,MappedStatement>
2. xml的映射文件，id不能重复
    - select:接受的参数parameterType| 数据返回的参数 resutlType
    - insert parameterType|useGenerateKeys|keyProperty 插入操作时，可以通过配置，返回插入后的数据库对应的字段值（一般为主键id）通常使用于 配置有自增主键时
    - update:parameterType
    - delete:parameterType
    - 接收动态参数:#{属性名|map的key}
3. 核心的三个对象
    - SqlSessionFactoryBuilder-->build(InputStream in)
    - SqlSessionFactory(DefaultSqlSessionFactory)
    - Configuration配置对象,配置文件与映射文件在内存中的对象.
    - SqlSession(DefaultSqlSession)-->selectOne|selectList|insert|update|delete|getMapper.
4. junit的使用
    - @FixMethodOrder固定多个测试方法的执行顺序

- @BeforeClass注解静态方法,全局初始化方法
- @Before注解测试初始化方法
- @Test注解测试方法
- @After注解测试销毁方法
- @AfterClass注解静态方法,全局销毁方法.

实体类中的属性不能用long类型的数据，因为转为json后发送给前端后，可能会丢失精度，因此一般用integer或者String。

1. ## #{}与${}的区别

   - 相同点：都是用来获取动态参数
   - 不同点：
     - #{}：会转为？占位符 preparedStatement
     - ${}: 直接获取动态参数 拼接为sql语句，Statement，不建议使用，一般用来做排序
     - 动态注入，一般在前端有输入框的情况下才会发生
2. ## 映射文件中的sql标签

   提取出公共的sql语句，通过include来引入sql语句块

```
  <sql id="basicFields">
    uid,uname,uemail,uphone,uwechat
  </sql>

  <sql id="allFields">

uid,uname,uemail,uphone,uwechat,create_time,update_time,create_by
  </sql>

  <sql id="basicQuery">
    select uid,uname,uemail,uphone,uwechat,create_time,update_time,create_by from sysuser
  </sql>

  <select id="selectUsers" resultType="sysuser">
    select
```

```
    <include refid="basicFields"></include>
    from sysuser order by ${by} ${orderType}
  </select>
```

3. resultMap标签（重要）

resultMap: 结果集映射标签 把查询结果列映射到指定类的指定成员变量上

结果集映射标签(建议配置所有的字段的映射关系)

在select标签中结果集的类型设置为resultMap=""

resultMap标签中，可以配置映射关系，就是数据库的字段名和实体类中的属性名字不一致时，可以使用该映射，让他们一一对应起来。column是指数据库的字段名，proterty是指实体类的属性名

```
主键映射：<id column="uid" property="uid"></id>
非主键映射：<result column="uname" property="uname">
</result>
```

```
  <resultMap id="sysuserResultMap" type="sysuser">
    <id column="uid" property="uid"></id><!--主键列映射-->
    <result column="uname" property="uname2">
</result>
    <result column="uemail" property="uemail">
</result>
    <!--建议所有字段都映射上-->
  </resultMap>

  <select id="selectByKey" parameterType="int"
resultMap="sysuserResultMap">
    <include refid="basicQuery"></include>
    where uid=#{uid}
  </select>
```

注意点：select标签的resultMap属性不能与resultType同时出现

一个映射文件可能有多个resultMap标签

4. 对象关系映射-多对一映射| 持有关系映射（重要）

数据库表的设计关系：1对1 1对多 多对1 多对多

1-N ： 在1方关联多方的主键作为外键。

N-N： 使用中间表来进行关联两个表的主键作为外键

对象层面的设计关系：持有关系和聚合关系 建议看《uml模型设计》书籍

```
A{
  B b; //持有关系
  List<C> cs;//聚合关系
}
```

实现关联查询一共有三种方法

方法1：手工进行两次单表查询，并将查询出来的数据进行拼接，(一般在一的表所对应的实体类(a)中，添加一个多的表所对应的实体类(b)对象作为a实体类的一个成员变量)

方法2：使用sql语句进行表的链接查询，一次查询出所有的记录，然后分别进行映射，适用于查询列表

```
  <resultMap id="userAndRoleMap" type="sysuser">
    <id column="uid" property="uid"></id>
    <result column="uname" property="uname2">
</result>
    <result column="upwd" property="upwd"></result>
    <result column="uphone" property="uphone">
</result>
    <result column="uwechat" property="uwechat">
</result>
    <result column="uemail" property="uemail">
</result>
    <result column="create_time"
property="createTime"></result>
    <result column="update_time"
property="updateTime"></result>
    <result column="create_by" property="createBy">
</result>
```

```
    <!--sysuser类中的srole成员变量映射-->
    <association property="srole"
javaType="Sysrole">
      <id column="rid" property="rid"></id>
      <result column="rname" property="rname">
</result>
      <result column="rdesc" property="rdesc">
</result>
      <result column="rctime"
property="createTime"></result>
      <result column="rutime"
property="updateTime"></result>
    </association>
  </resultMap>

  <select id="selectUserAndRoleByUid"
parameterType="int" resultMap="userAndRoleMap">
    select u.*,r.rname,r.rdesc,r.create_time as
rctime,r.update_time as rutime from sysuser u left
JOIN sysrole r on u.rid=r.rid where u.uid=#{uid}
  </select>
```

方法3:使用mybatis内部的二次查询操作.(了解)

```
  <resultMap id="userAndRoleMap2" type="sysuser">
    <id column="uid" property="uid"></id>
    <result column="uname" property="uname2">
</result>
    <!--
              column:二次查询需要的参数列
              property:sysuser类中的成员变量名
              javaType:类型
              select:二次查询的位置
          -->
    <association column="rid" property="srole"
javaType="Sysrole"
select="com.javasm.mapper.SysroleMapper.selectRoleB
yKey"></association>
  </resultMap>
  <select id="selectUserAndRoleById2"
parameterType="int" resultMap="userAndRoleMap2">
```

```
    select * from sysuser where uid=#{uid}
  </select>
```

设计表的字段必须写注释信息---->提高代码的可阅读性

使用场景： 查询用户的列表(用户名 手机号 角色名)

5. 对象关系映射-1对多映射 | 聚合关系映射（重要）

应用场景：查询角色时，查询该角色下的所有用户

聚合应该用 collection标签，需指定ofType（集合的范型）

方法1:手工进行多次单标查询,把查询结果组合

```
  SysuserMapper um =
session.getMapper(SysuserMapper.class);
  SysroleMapper rm =
session.getMapper(SysroleMapper.class);
  int rid=2;
  //第一次查询:单查角色表
  Sysrole sysrole = rm.selectRoleByKey(rid);

  //第二次查询:单差用户表
  List<Sysuser> users = um.selectUsersByRoleId(rid);
    sysrole.setUsers(users);
```

方法2:使用sql表链接查询

```
  <resultMap id="roleAndUsersMap" type="sysrole">
    <id column="rid" property="rid"></id>
    <result column="rname" property="rname">
</result>
    <result column="rdesc" property="rdesc">
</result>
    <collection property="users" ofType="Sysuser">
<!--List<Sysuser>-->
      <id column="uid" property="uid"></id>
      <result column="uname" property="uname2">
</result>
      <result column="upwd" property="upwd">
</result>
```

```
      <result column="uphone" property="uphone"></result>
      <result column="uwechat" property="uwechat"></result>
      <result column="uemail" property="uemail"></result>
    </collection>
  </resultMap>

  <select id="selectRoleAndUsersByKey" parameterType="int" resultMap="roleAndUsersMap">
    select r.rid,r.rname,r.rdesc,u.uid,u.uname,u.upwd,u.uphone,u.uwechat,u.uemail from sysrole r left join sysuser u on r.rid=u.rid where r.rid=#{rid}
  </select>
```

方法3:mybatis内部发起二次单表查询,不建议使用

```
  <resultMap id="roleAndUsersMap2" type="sysrole">
    <id column="rid" property="rid"></id>
    <result column="rname" property="rname"></result>
    <result column="rdesc" property="rdesc"></result>
    <collection property="users" ofType="Sysuser" column="rid" select="com.javasm.mapper.SysuserMapper.selectUsersByRoleId"></collection>
  </resultMap>

  <select id="selectRoleAndUsersByKey2" parameterType="int" resultMap="roleAndUsersMap2">
    select * from sysrole where rid=#{rid}
  </select>
```

6. 动态sql语句（重要）

非常重要的知识点

在xml映射文件中进行sql语句拼接的一种方式，使用where标签，可以生成sql中的where关键字，并忽略紧跟之后的and或者or，如果要使用模糊查询则必须用"%"或者使用concat函数进行拼接，如果where标签中的所有条件都部满足则不会生成where关键字.

- where
  - where ---if
  - 在xml映射文件中进行sql语句拼接的一种方式，使用where标签，可以生成sql中的where关键字，并忽略紧跟之后的and或者or
  - 如果要使用模糊查询则必须用"%"#{rname}"%"(必须使用双引号)或者使用concat函数进行拼接
  - 如果where标签中的所有条件都不满足则不会生成where关键字.
  - 如果要使用<号(会与标签的开始进行混淆)，则必须使用特殊符号 <，小于等于用 <=#{rid}
  - ```
    CONCAT("%",#{rname},"%")
    "%"#{uname}"%"

    小于比较,小于号存在歧义,需要使用特殊符号
    小于: and rid < #{rid}
    小于等于:and rid <= #{rid}
    ```
- if
  - 条件判断 test属性写boolean表达式
- set
  - 一般用于update标签中，可以生成set关键字
  - 生成set关键字，并忽略最后的一个逗号。rname=#{rnaem},
  - 不会把其他为null值 置为null，不会重置其他未修改的字段
- foreach
  - 循环标签：用于批量删除 批量添加 上面
  - foreach中有5个属性，collection="array"---->collection为默认封装的key的值，open="（",循环的开始标志 close=")"循环结束的标志，item="roleid"：循环变量的名字，separator=","循环变量之间使用，分隔。
    ```

        - mybatis底层把数组类型的参数，封装为Map，map.put("array",rid)
        - mybatis底层把集合类型的参数，封装为Map，map.put("list",rid)
        - 通过@Param 可以自定义封装后的key值（在接口定义的方法中使用该注解）
        - ----------使用添加操作----------
        - 使用对象的数组
    - choose(when,otherwise)
        - 类似于switch case，where-choose--when，只可以选择一个条件，如果有 一个条件满足，则其他的when都不会执行，如果都不满足则执行 otherwise

7. mybatis延迟加载

主要掌握延迟加载的思想理念，分页也是延迟加载，一般是为了用户体验，一般条数超过100的都需要延迟加载

图片是一般是游览器加载速度比较慢的

1. ```
   <!--全局开启延迟加载-->
   <setting name="lazyLoadingEnabled" value="true"></setting>

   <!--改积极加载为消极加载-->
   <setting name="aggressiveLazyLoading" value="false"></setting>
   <!--禁用toString,equals,hashcode,clone方法的触发延迟加载属性-->
   <setting name="lazyLoadTriggerMethods" value=""></setting>
   ```

2. 只存在于mybatis中使用二次查询的时候，即先查询用户，再查询角色的时候才会出现
3. 需要在配置文件中进行延迟加载的设置
4. 默认只做第一次的查询，当需要获取关联对象的时候，才做第二次的查询
   ```

5. 其中可以选择进行配置（aggressiveLazyLoading true|false），当为true时，当有调用有延迟加载的属性的对象时，就会进行二次加载，如果禁用(false)时，则只有加载需要延迟加载的对象时才会触发延迟加载。
6. lazyLoadTriggerMehods,延迟加载的触发方法，默认(toString,clone,hashcode,equals)时，可以进行value=""的配置

8. mybatis缓存使用

orm框架中都有缓存的实现，但是一般在项目中没有啥用。

一般在web开发中有专门的缓存的方法

一般不会使用，相同的语句，一般不会进行查询第二次

一级缓存：session级别的缓存，默认开启

- 查询过程：先去sqlSession对象的缓存空间查询数据
- 查询到则直接返回
- 如果查询不到，则查询数据，并将查询的结果保存到sqlSession中

二级缓存：默认是没有开启的，需要在setting中进行配置,跨会话共享空间，可以在不同会话之间共享缓存的数据，相当于全局的缓存

```
1.在配置文件中需要进行配置，mybatis.xml
2.需要在map的xml文件中进行配置
1.全局开启二级缓存
<setting name="cacheEnabled" value="true">
</setting>

2.map的xml中的配置
eviction缓存策略：先进先出
flushInterval: 缓存的刷新时间，设置为60s
size:最大的缓存的数量
readonly:只读（无法修改缓存空间中的数据）
<cache eviction="FIFO" flushInterval="6000"
size="512" readOnly="true"/>
// 如果执行了insert或者update操作则可以指定flushCache：可以刷新缓存，可以把已有的缓存清空，强制刷新缓存
<update flushCache="true">

以上配置生效后，该配置文件中的所有查询操作都默认使用缓存的操作。如果某个查询操作不使用二级缓存，则可以进行设置
```

```
16  <select useCache="false"></select>
```

9. # 代理模式-静态代理实现

   23种设计模式

   可以了解disruptor高并发框架，被很多高并发采用。

   代理(proxy)模式的使用场景：当一个已经存在对象，某个方法不满足需求时，使用代理模式，代理模式分为：静态代理和动态代理

   静态代理是基础：使用实现或者继承创建一个新的代理类，重写需要修改的方法

   代理类与被代理类同类型

   代理类持有被代理（被代理的对象作为代理类的一个属性）

   经典应用场景：Connection------->close方法关闭连接

   DruidDataSource---->Connection conn = getConnection()---->conn.close() 方法可以放回连接池中

   就是dataSource 作为connection的一个代理，修改了conn中的close方法

10. # 总结与重点

    resultMap标签 association（持有关系）与collection（聚合关系）与id和result的四个标签

    动态sql：if|where|set|foreach

    静态代理