MyBatis

1.三层架构

1.界面层：用来做用户的请求，调用业务逻辑层的处理，并返回处理结果给客户端

2.业务逻辑层：负责对数据的业务逻辑，并了调用数据访问层，并且返回处理结果给界面层，事务控处理方式这层

3.数据访问层：负责对有数据库中的增删改查的操作，并且为业务逻辑层提供数据

4.调用顺序：界面层 ---> 业务逻辑层 ---> 数据访问层

2.SSM框架

1.Mybatis框架：优秀的持久层的框架，简化数据库的访问，用了对对象的访问取代原来的SQL

2.Spring框架：整合框架，是一个容器，是一个全家桶，提供各种问题的解决方案，核心技术IOC和AOP

3.SpringMVC框架：完成控制器的优化，数据绑定，数据，数据封装提供了简单高效的处理

3.MyBatis框架的访问

- 1.简单访问方式
  - 1.访问步骤
    - 1.新建数据库，建表，表结构初始化
    - 2.新建Maven工程，进行目录 修改，pom.xml文件的修改
      - 1.添加依赖
      - <!-- 原版mybatis的依赖 -->
        - <dependency>
        - <groupId>org.mybatis</groupId>
        - <artifactId>mybatis</artifactId>
        - <version>3.5.1</version>
        - </dependency>
      - <!-- 连接数据库驱动的依赖 -->
        - <dependency>
        - <groupId>mysql</groupId>
        - <artifactId>mysql-connector-java</artifactId>
        - <version>8.0.22</version>
        - </dependency>
        - </dependencies>>
      - 4.指定资源位置：
        - <build>
        - <resources>
        - <resource>
        - <directory>src/main
        - /resources</directory>
        - <filtering>false
        - </filtering>false</filtering>
        - <includes>**/*.anti;</include>
        - <includes>**/*.properties</include>
        - <includes>
        - <filtering>false</filtering>false</filtering>
        - </resource>
        - </resources>
        - </build>
    - 5.拷贝db.properties文件到resources目录下
    - 6.新建SqlMapConfig.xml文件，注意文件头的添加（从pdf帮助文件中找），完成Mybatis框架的核心配置
    - 7.新建实体类
    - 8.新建StudentDaoImpl.xml文件，注意文件的添加（从pdf帮助文件中找），完成数据库的增删改查的操作
    - 9.完成测试功能：
      - @Before:在所有测试方法执行前先执行这行注解中的代码
      - @After:在所有测试方法执行结束后执行这行注解中的代码
      - API文档:SqlSession
      - 功能：
        - //创建文件流，读取配置文件中的配置信息
        - InputStream inputStream = Resources.getResourceAsStream("SqlMapConfig.xml");
        - //创建工厂构造器对象
        - SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(inputStream);
        - //从工厂中得到SqlSession对象，它相当于是Connection+PreparedStatement
        - session = factory.openSession();
        - //执行数据库的操作
        - List<Student> list = session.selectList("com.bjpowernode.student.getAll");
        - //显示结果集合
        - list.forEach(this -> System.out.println(n));
- 2.动态代理访问
  - 1.新建表结构
  - 2.新建maven项目，修改目录，修改pom文件
  - 3.添加依赖，添加资源文件的位置指定
  - 4.配置db.properties文件到resources目录下
  - 5.添加SqlMapConfig.xml核心配置文件，并开发
  - 6.编写实体类
  - 7.新建com.bjpowernode.mapper包
  - 8.新建UsersMapper.java接口
  - 9.新建UsersMapper.xml文件
  - 10.在UsersMapper.xml文件开发功能：增删改查
  - 11.测试功能
- 3.增删改查
  - 1.UsersMapper接口写UsersMapper.xml文件必须信息一致
  - 2.UsersMapper接口写UsersMapper.xml文件表名必须一致，且唯不变
  - 3.UsersMapper接口中方法名称与UsersMapper.xml文件中的id必须一致
  - 4.UsersMapper接口中方法参数写UsersMapper.xml文件中parameterType类型应该一致
  - 5.UsersMapper接口中方法返回值写UsersMapper.xml文件中resultType类型应该一致（主要是查询操作）
  - 6.UsersMapper.xml文件中的namespace的值是UsersMapper接口的完全限定名称
  - 7.在SqlMapConfig.xml文件中注册UsersMapper.xml文件，使用<mappers>接口的完全限定名称
- 4.SqlMapConfig.xml文件的优化
  - 1.注册别名
    - 1.单个别名注册
      - <!-- 为每个实体起别名，一个一个起 -->
      - <typeAlias type="com.bjpowernode.pojo.Student" alias="student" />
      - </typeAlias>
    - 2.批量注册
      - <!-- 批量起别名，起名字的时候有规，前一起起别名，别名是：类名首字母小写 -->
      - <package name="com.bjpowernode.pojo" />
  - 2.设置日志输出
    - <!-- 打印日志输出sql语句的设置 -->
    - <settings>
    - <setting name="logImpl" value="STDOUT_LOGGING"/>
    - </settings>>
  - 3.注册mapper.xml文件
    - 1. 第一种简单的方式不可扫注册
      - <mapper resource="StudentDaoImpl.xml">
      - </mapper>
    - 2.动态代理方式可以注册
      - <mapper class="com.bjpowernode.mapper.UsersMapper">
      - </mapper>
    - 3.批量注册
      - <package name="com.bjpowernode.mapper"/>
- 5.Sql语句的编辑
  - 1.传入参数
    - 1.可以直接写单个参数的参数，不用考虑参数的名称
    - 2.如果传入的参数是多个参数，则必须使用#{arg0}或#{arg1}
    - 3.如果是复杂的参数，则可以使用#{arg0.属性名}
    - 4.如果是多个参数，则可以使用#{param1.属性名}
  - 2.返回结果
    - 1.如果是简单的数据类型，则可以使用#{属性名}
    - 2.如果是复杂的数据类型，则必须使用#{属性名.属性名}
    - 3.如果是多个参数，则可以使用#{param1.属性名}
  - 3.主键策略
    - 1.自增：在一个业务中有多条数据的时，下一个操作会用到上一个操作生成的主键信息
      - select LAST_INSERT_ID()
      - 说明：
      - <selectKey resultType="int" order="AFTER" keyProperty="id">
      - select LAST_INSERT_ID()
      - </selectKey>
      - 参数说明：
      - resultType：设置返回的主键的类型
      - order:生成主键的方法与入参的主键关之后
      - keyProperty：实体类中的主键变量的名称
    - 2.UUID：
      - select uuid()
      - 说明：
      - <selectKey resultType="string" order="BEFORE" keyProperty="orderId">
      - select UUID()
      - </selectKey>
      - 参数说明同上
- 6.结果集
  - 1.<sql>：代码片断，提取公用使用的代码用作组成
    - 示例：
    - <sql id="columns">，id,username,birthday,sex,address</sql>
  - 2.<include>引用代码片断
    - 示例：
    - select
      - <include refid="columns">
      - </include>
      - from users
      - where username like '%$username%';
  - 3.<if>条件判断
    - <if test="username != null">
      - and username like '%$username%'
      - </if>
    - 注意：test的参数和函数中的成员变量名称
  - 4.<where>条件判断
    - <select id="getByCondition" resultType="users" parameterType="users">
      - select <include refid="columns">,</include>
      - from users
      - <where>
      - <if test="address != null">
      - and address like '%$address%'
      - </if>
      - </where>
      - </select>>
  - 5.<set>有条件的更新，但是注意至少要加一列，这时方式和第3条条件的结果是没有相信，则数据库中相应的字段不要改
    - <update id="updateCondition" parameterType="users">
      - update users
      - <set>
      - <if test="userName != null">
      - username=#{userName},
      - </if>
      - ,,,,,,
      - </set>>
      - where id=#{id}
      - </update>>
  - 6.<foreach>:表单进行循环遍历
    - <foreach collection="list" item="str" separator=",", open="(" close=")">
      - #{str}
      - </foreach>>
    - collection：数组传入的内容，如果是List集合，则填写list,如果是数组，则填写array,如果是Map，则填写map
    - item：：表示迭代表的数据变量的别名
    - separator：多个值之间的分隔符
    - open：：整个循环语句最前面的符号
    - close：：整个循环语句最后的符号
  - 7.批量删除
    - <delete id="deleteBatch">>
      - delete from users where id in
      - <foreach collection="array" separator=",", item="id" open="(" close=")">
      - #{id}
      - </foreach>>
      - </delete>>
  - 8.批量增加
    - <insert id="insertBatch">>
      - insert into users (username,birthday,sex,address)
      - values
      - <foreach collection="list" separator=",", item="user">
      - (#{user.userName},#{user.birthday},#{user.sex},#{user.address})
      - </foreach>>
      - </insert>>
  - 9.回显参数位置：如果参数不是基本类或包装变量，是数的条件，可以使用指定下标位置的方式进行传参
    - 测试例：
    - List <Users> getByLocationDate begin, Date end);
    - 示例：
    - <select id="getByLocation" resultType="users">
      - select <include refid="columns">,</include>
      - from users
      - where birthday between #{arg0} and #{arg1}
      - </select>>
  - 10.入参是map，当直接为传入的条件实体类无法封装，使用map的形参传参，使用键值对的方式来传参。
    - 测试例：
    - List <Users> getByLocation(Map map); map.put("arg0",begin); map.put("arg1",end);
    - 说明：
    - <select id="getByLocation" resultType="users">
      - select <include refid="columns">,</include>
      - from users
      - where birthday between #{arg0} and #{arg1}
      - </select>>
    - 说明：#{arg0}是map的键值
    - 测试例：
    - Map<String,Date> map = new HashMap<>();
      - map.put("arg0","yyyy-MM-dd").parse("1996-01-01");
      - map.put("arg1",new SimpleDateFormat("yyyy-MM-dd").parse("1996-01-03"));
  - 11.返回结果map：可能会出现返回的结果不是实体类的情况，一般用于多张表的联合查询
    - 说明同resultMap的情况一样
    - map的键是列的名称或别名
    - map的值是列中的内容
    - 示例：
    - 测试例：
      - Map<String,Object> getReturnMapOneInt(int id);
      - 测试例：
      - 说明：返回结果map,并且只能一条数据 -->
      - <select id="getReturnMapOne" parameterType="int" resultType="map">
      - select id myId,username myname,birthday mybirthday,address myaddress,sex mysex
      - from users
      - where id=#{id}
      - </select>>
- 7.resultMap
  - 功能：进行列值与属性变量名称不一致的情况时，还有如果多个实体类时使用，也要通过resultMap进行关联映射
    - 1.相同部分：让部分与成员变量名称一致
      - 2. <resultMap id="booknap" type="book">
      - <id property="id" column="bookid" />
      - <result property="name" column="bookname">
      - </result>
      - 说明：
      - 参数解析：
      - result：绑定非主键列
      - id:绑定主键列值
    - 2.<resultMap id="bookmap" type="book">
      - <id property="id" column="bookid" />
      - <result property="name" column="bookname">
      - </result>
      - 参数解析：
      - result：绑定非主键列
      - id:绑定主键列值
  - 1.一对多
    - 一对多，是指的是表产生记录，一个表产生多个记录
    - 一对多使用 collection标签
    - 三个关键属性及值是：
    - public class Customer{
    - private int id; ----一对多的数据是一个客户的信息，对应的是一个客户的所有订单信息
    - property: 表中的实体变量的名称
    - ofType: 表示集合中的数据类型
    - 说明：
      - private String name;
      - private int age;
      - 设计一个订单的集合形式的集合
      - private List <Orders> orderList;
    - 示例：
      - <resultMap id="customermap" type="customer">
    - <!-- 绑定实体类中普通的成员变量 -->
      - <id property="id" column="id" />id>
      - <result property="name" column="name" />result>
      - <result property="age" column="age" />result>
    - <!-- 绑定实体类中集合类型的成员变量 -->
      - <collection property="ordersList" ofType="orders" >
      - <id property="id" column="oid" />id>
      - <result property="orderNumber" column="orderNumber" />result>
      - <result property="orderPrice" column="orderPrice" />result>
      - </collection>>
      - </resultMap>>
  - 2.多对一
    - 多对一的情况，订单和客户相关多对一，每个订单都有与之关联的一个客户
    - 使用<association>标签
    - public class Orders{
      - private int id;
      - private String orderNumber;
      - private double orderPrice;
      - //多方持有一方的对象
      - //形式是属于自己等的客户关联关系，？ list 和 持有对象 OK
      - private Customer customer;
    - 说明：
    - <!-- 因为实体类Orders中包含Customer的属，多一所表示表类产的信息时，必须使用resultMap进行绑定
    - -->
      - <resultMap id="ordersmap" type="orders">
      - <id property="id" column="id" />id>
      - <result property="orderNumber" column="orderNumber" />result>
      - <result property="orderPrice" column="orderPrice" />result>
    - <!-- 解析Customer成员变量 -->
      - <association property="customer" javaType="customer" >
      - <result property="id" column="cid" />result>
      - <result property="name" column="name" />result>
      - <result property="age" column="age" />result>
      - </association>>
      - </resultMap>>
  - 3.一对一
    - 在实现的表中持有本表的对象，在表中持有数据的时，表现在表现的时 表 有的信息
    - 在表的数据中持有实体的对象，在表中持有数据的时，表现在表现的时 表 有的信息
    - 持有对象使用标签 <association> 解析关系的属
    - 在实现UsersDao的文件中 根据就是表的主键（主键实现）
    - <select id="getAllUsersAndTeacher" resultMap="teacher" >
      - <id property="id" column="id" />id>
      - <result property="name" column="name" />result>
      - </resultMap>>
    - 4.多对多
    - 一多 和多多 个表的，一个表的下有多本书，所以书和表的之间是多对多的关联关系。 ex解多对多关联关系是通过中间表来解决，中间表是图书表和表的表的外键值。
    - 在实现UsersDao的文件中
      - <collection property="cfoc" ofType="category" >
      - <id property="id" column="cid" />id>
      - <result property="name" column="name" />result>
      - </collection>>
    - 1. 总结：
    - 用法解什么关联关系，如果某对持有另一方的集合，则使用<collection>标签实现解析，如果某对持有另一方的对象，则使用<association>标签实现解析。
- 8.关联关系
  - 1.数据库表的关系
    - 1.1.表的关联的体现：不是在数据关联，
    - 2.1.关联关系表的：一对多，多对一，一对一，多对多
    - 3.一对多关系实现有两方面的
  - 2.数据库的表的关系
    - 2.数据库的表的关系：是指一个表的数据，在其他表的数据的时，表现在表现的时 表 有的信息
    - 在表的数据中持有其他表的对象，在表中持有数据的时，表现在表现的时 表 有的信息
    - 持有对象使用标签 <association> 解析关系的属
    - 在实现UsersDao的文件中 根据就是表的主键（主键实现）
    - <select id="getAllUsersAndTeacher" resultMap="teacher" >
      - <id property="id" column="id" />id>
      - <result property="name" column="name" />result>
      - </resultMap>>
    - 4.多对多
    - 一多 和多多 个表的，一个表的下有多本书，所以书和表的之间是多对多的关联关系。 ex解多对多关联关系是通过中间表来解决，中间表是图书表和表的表的外键值。
    - 在实现UsersDao的文件中
      - <collection property="cfoc" ofType="category" >
      - <id property="id" column="cid" />id>
      - <result property="name" column="name" />result>
      - </collection>>
    - 1. 总结：
    - 用法解什么关联关系，如果某对持有另一方的集合，则使用<collection>标签实现解析，如果某对持有另一方的对象，则使用<association>标签实现解析。
- 9.缓存
  - 1.MyBatis的数据的事务处理模式是JDBC，是手工提交事务的方式。为简化调用session的时候，默认就是手工提交。但可以调整为。
    - session = factory.openSession(true);//true代表自动提交
  - 2.MyBatis 支持两种事务管理器类型：JDBC 与 MANAGED，MANAGED是容器管理的事务处理模式
- 10.缓存
  - 1.目的：缓存是指的目的是提高查询效率。
  - 2.缓存的目的是：如查询数据的结果时，先去缓存查一下，如果有，直接返回结不的结果集，如果没有的时候数据库的时，查到数据，在缓存放一份，再返回客户端。 第一次查询时，直接查询数据库，当数据库中的数据发生变化时，则清空缓存。
  - 3.一级缓存，自动开启，如果数据库中发生增删改操作时，缓存清空。SqlSession实现的，清空缓存。
  - 4.二级缓存，手工开启，如果数据库中发生增删改操作时，缓存清空。
    - 二级缓存开启的步骤
      - 1).在SqlMapConfig.xml文件中添加开启二级缓存配置
      - <setting name="cacheEnabled" value="true">
      - 2).在指定的mapper.xml文件中添加<cache>
      - </cache>
      - 3).实体类的pojo文件实现Serializable接口
- 11.ORM
  - 1.ORM:Object Relational Mapping 对象关系映射
  - object:java语中的对象
  - Relational：数据库中的关系表（数据库表）
  - Mapping：映射：将对java语中的对象与数据库表中的
  - ??？？？？
  - 3.MyBatis框架就是ORM框架的一个框架，又称为半ORM的优秀的框架
  - 4.优点：从为开发者提供数据操作
    - 1.数据库中的数据的操作
    - 这个过程的为的为元
- 12.测试
  - 测试用