1. MyBatis的DAO层实现
2. 传统方式

就是MVC思想的开发模式，只不过将MyBatis的使用放到DAO层的实现类中。

1. 创建DAO层的接口和实现类

接口：

package dao;  
  
import domain.User;  
  
import java.io.IOException;  
import java.util.List;  
  
public interface UserMapper {  
 public List<User> findAll() throws IOException;  
}

实现类：

package dao.impl;  
  
import dao.UserMapper;  
import domain.User;  
import org.apache.ibatis.io.Resources;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.List;  
  
public class UserMapperImpl implements UserMapper {  
 @Override  
 public List<User> findAll() throws IOException {  
 InputStream resourceAsStream = Resources.*getResourceAsStream*("sqlMapConfig.xml");  
 SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(resourceAsStream);  
 SqlSession sqlSession = sqlSessionFactory.openSession(true);  
 List<User> userList = sqlSession.selectList("userMapper.findALl");  
 return userList;  
 }  
}

1. 在Service层中调用DAO层的实现类

package service;  
  
import dao.UserMapper;  
import dao.impl.UserMapperImpl;  
import domain.User;  
  
import java.io.IOException;  
import java.util.List;  
  
public class ServiceTest {  
  
 public static void main(String[] args) throws IOException {  
 UserMapperImpl userMapper = new UserMapperImpl();  
 List<User> all = userMapper.findAll();  
 System.*out*.println(all);  
  
 }  
}

1. 代理开发方式
2. 代理开发方式规范

MyBatis实现DAO层的开发，可以使用代理开发的方式。

Mapper接口开发方法只需要我们编写Mapper接口（相当于DAO接口），再由MyBatis框架根据接口的定义创建接口的动态代理对象。想要让MyBatis框架自动创建代理对象，那么Mapper接口开发就要遵循一定的规范：

1. 映射文件中的namespace与Mapper接口的全限定名相同

映射文件中每个mapper标签都有属性namespace，每个mapper对应一个Mapper接口，要想通过接口动态地使用映射文件中的SQL语句，那么对应的映射文件中的mapper标签的namespace就要改为Mapper接口的全限定名。

1. Mapper接口方法名和映射文件中定义的每个statement的id相同

Mapper接口中的每个方法对应映射文件中的一个sql语句，而每个sql语句都由一个id对应，因此每个id应和Mapper接口中的对应方法名相同。

1. Mapper接口方法的输入参数类型和映射文件中定义的每个sql语句的parameterType属性的类型相同

Mapper接口中的方法的传入参数代表着要使用该参数，对应于映射文件中的sql语句就是要抽取实例对象中的参数，因此在映射文件中我们传入的是实例对象的类，使用parameterType属性来表示该实例对象。因此，Mapper接口方法的输入参数类型要和映射文件中的每个sql语句的parameterType的类型相同。

1. Mapper接口方法的输出参数类型和映射文件中定义的每个sql的resultType的类型相同

同理，映射文件中sql语句的resutlType属性代表着要将数据封装成一个什么类型的对象，对应于Mapper接口方法，即输出什么类型的参数。

1. 使用代理方式

当我们定义好了Mapper接口和对应的映射文件后，我们就可以来使用这种代理方式了，创建sqlSession的操作都是一样的，然后通过sqlSession对象的getMapper(Mapper.class)方法生成Mapper接口的代理对象，传入的参数为Mapper接口的字节码，之后就可以通过这个代理对象来调用各种自定义的接口中的方法了。

UserMapper userMapper = sqlSession.getMapper(UserMapper.class);

User user = userMapper.findById(1);

1. MyBatis的动态SQL语句

之前我们是将SQL语句硬编码在映射文件中，这在业务场景中是不合理的，使用动态地SQL语句是最高效的。如在id不为空时根据id查询，在username不为空时根据username查询。这种情况只有动态的SQL语句才能够实现。

1. <if>标签

在映射文件中，我们要将SQL语句的硬编码改为动态的SQL，因此就要使用条件判断。<if>标签代表着如果的意思，其中的test属性表示测试。if标签的目的是判断条件，然后将符合条件的分支添加到SQL语句中，前面还有一个<where>标签。

如，

<if test=”id!=0”>

and id=#{id}

</if>

表示如果id!=0，则id=#{id}

1. foreach标签

foreach标签用于遍历集合，collection属性表示集合的类型，open属性表示动态sql语句的开始字段，close属性表示动态SQL语句的结束字段，item属性表示集合中每一项的名称， separator属性表示集合中的每项放到动态SQL语句中应该用什么符号分隔。

如：

<select id=”findByIds” parameterType=”list” resultType=”user”>

select \* from user

<where>

<foreach collection=”list” open=”id in(” close=”)” item=”id” separator=”,”>

#{id}

</foreach>

</where>

</select>

这段动态SQL语句表示传入的是一个集合元素，动态SQL语句的起始为 “id in(”，结束为”)”，每一项都是id，每一项在动态SQL中用”,”分隔开。

1. SQL片段抽取

在业务中我们通常会写大量的重复SQL语句，为了减少工作量，可以将重复的SQL语句抽取出来，达到SQL重用的目的。

sql标签抽取SQL语句，id属性为标识：

<sql id=”selectUser”> select \* from user</sql>

当我们要使用这一字段时，通过include标签即可引用，将include标签代替原有的SQL语句即可，refid属性为该字段的id：

<include refid=”selectUser”></include>

1. typeHandlers标签
2. 用途

无论是MyBatis在预处理语句中设置一个参数，还是从结果集中取出一个值，都会用类型处理器将获取的值以合适的方式转换成java类型。

MyBatis默认有一些类型处理器，如BooleanTypeHandler，ByteTypeHandler，但是很多类型是没有默认的类型处理器的，需要我们自己来创建。

1. 创建类型处理器

实现org.apache.ibatis.type.TypeHandler接口，或者继承类org.apache.ibatis.type.BaseTypeHandler，然后可以选择性地将它映射到一个JDBC类型中。

1. 实例

一个Java中的Date数据类型，将它存到数据库时存成一个毫秒数，取出来时转换成Java的Date，即Java的Date与数据库的varchar毫秒数之间的转换。

步骤：

1. 定义转换类继承类BaseTypeHandler<T>
2. 覆盖4个未实现的方法，其中setNonNullParameter为Java程序设置数据到数据库的回调方法，getNullableResult为查询时mysql的字符串类型转换为java的Type类型的方法。
3. 在MyBatis核心配置文件中进行注册。在MyBatis的核心配置文件中注册类型处理器。<typeHandlers>标签代表多个类型处理器，<typeHandler>标签着一个类型处理器。属性handler为转换类的全限定名。

<typeHandlers>

<typeHandler handler=”com.mybatis.handler.DateTypeHandler”></typeHandler>

</typeHandlers>

1. plugins标签

plugins即插件，MyBatis可以使用第三方的插件来对功能进行拓展。如分页助手PageHelper是将分页的复杂操作进行封装，使用简单的方式即可获得分页的相关数据。

步骤：

1. 导入通用PageHelper的坐标

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>3.7.5</version>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

<version>0.9.1</version>

</dependency>

1. 在MyBatis核心配置文件中配置PageHelper插件

在MyBatis的核心配置文件中，我们设置了plugins标签，其中可以添加多个plugin子标签，每个标签代表着一个插件，interceptor属性表示着该插件的全限定名。还可以为该插件设置参数，如dialcet方言参数，设置为mysql。因为各种sql语句的不同语法会有差异，因此设置为所使用的的数据库软件。

<plugins>

<plugin interceptor=”com.github.pagehelper.PageHelper”>

<property name=”dialect” value=”mysql”></property>

</plugin>

</plugins>