MyBatis

一对一、一对多、多对多

> 单表查询

```
public Book findById(Integer id);
```

```
<select id="findById" parameterType="java.lang.Integer"
resultType="com.southwind.entity.Book">
    select * from book where bookid = #{id}
</select>
```

1、在接口中定义方法

2、在 Mapper.xml 中定义方法对应的 SQL，以及解析的模板

id 对应方法名

parameterType 对应参数类型

resultType 对应结果类型

> parameterType

支持基本数据类型、包装类、String、多参数

```
public Book findByNameAndPrice(String name,Double price);
```

```
<select id="findByNameAndPrice" resultType="com.southwind.entity.Book">
    select * from book where name = #{param1} and price = #{param2}
</select>
```

多个参数通过参数下标进行映射，param1、param2、param3.....

> resultType

支持集合、实体类、基本数据类型、包装类、String

```
public Integer count();
```

```
<select id="count" resultType="java.lang.Integer">
    select count(*) from book
</select>
```

```
public String findNameById(Integer id);
```

```
<select id="findNameById" resultType="java.lang.String">
    select name from book where bookid = #{id}
</select>
```

> 多表关联查询

一对多

```
package com.southwind.mapper;

import com.southwind.entity.Student;

public interface StudentMapper {
    public Student findById(Integer id);
}
```

```
<?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.southwind.mapper.StudentMapper">

    <resultMap id="studentMap" type="com.southwind.entity.Student">
        <id property="id" column="id"></id>
        <result column="name" property="name"></result>
        <result column="age" property="age"></result>
        <association property="classes" javaType="com.southwind.entity.Classes">
            <id column="cid" property="id"></id>
            <result column="cname" property="name"></result>
        </association>
    </resultMap>

    <select id="findById" resultMap="studentMap">
        select s.id id,s.name name,s.age age,c.id cid,c.name cname from classes
c,student s where c.id = s.cid and s.id = #{id}
    </select>

</mapper>
```

```
package com.southwind.mapper;

import com.southwind.entity.Classes;

public interface ClassesMapper {
    public Classes findById(Integer id);
}
```

```
<?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.southwind.mapper.ClassesMapper">

    <resultMap id="classesMap" type="com.southwind.entity.Classes">
        <id column="cid" property="id"></id>
        <result column="cname" property="name"></result>
        <collection property="students" ofType="com.southwind.entity.Student">
            <id column="sid" property="id"></id>
            <result column="sname" property="name"></result>
            <result column="sage" property="age"></result>
        </collection>
```

```
    </resultMap>

    <select id="findById" resultMap="classesMap">
        select c.id cid,c.name cname,s.id sid,s.name sname,s.age sage
        from classes c,student s where c.id = s.cid and c.id = #{id}
    </select>

</mapper>
```

多对多

```
package com.southwind.entity;

import lombok.Data;

import java.util.List;

@Data
public class Account {
    private Integer id;
    private String name;
    private List<Course> courses;
}
```

```
package com.southwind.entity;

import lombok.Data;

import java.util.List;

@Data
public class Course {
    private Integer id;
    private String name;
    private List<Account> accounts;
}
```

```
package com.southwind.mapper;

import com.southwind.entity.Account;

public interface AccountMapper {
    public Account findById(Integer id);
}
```

```
<?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.southwind.mapper.AccountMapper">

    <resultMap id="accountMap" type="com.southwind.entity.Account">
        <id column="aid" property="id"></id>
        <result column="aname" property="name"></result>
        <collection property="courses" ofType="com.southwind.entity.Course">
            <id column="cid" property="id"></id>
```

```
            <result column="cname" property="name"></result>
        </collection>
    </resultMap>

    <select id="findById" resultMap="accountMap">
        select a.id aid,a.name aname,c.id cid,c.name cname from account a,course
c,account_course ac
        where a.id = ac.aid and c.id = ac.cid and a.id = #{id};
    </select>

</mapper>
```

```
package com.southwind.mapper;

import com.southwind.entity.Course;

public interface CourseMapper {
    public Course findById(Integer id);
}
```

```
<?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.southwind.mapper.CourseMapper">

    <resultMap id="courseMap" type="com.southwind.entity.Course">
        <id column="cid" property="id"></id>
        <result column="cname" property="name"></result>
        <collection property="accounts" ofType="com.southwind.entity.Account">
            <id column="aid" property="id"></id>
            <result column="aname" property="name"></result>
        </collection>
    </resultMap>

    <select id="findById" resultMap="courseMap">
        select a.id aid,a.name aname,c.id cid,c.name cname from account a,course
c,account_course ac
        where a.id = ac.aid and c.id = ac.cid and c.id = #{id};
    </select>

</mapper>
```

2.MyBatis延迟加载（争对于多表）

一种提高查询效率的方式，在数据有级联的情况下，可以动态地进行选择查询符合条件的结果。有主从关系的时候可以动态的根据条件去选择只查主表还是主从表一起查

延迟加载 又叫懒加载

User

```
@Data
public class User {
    private Integer id;
    private String username;
    private String password;
    private String birthday;
    private List<Orders> orders;


}
```

Orders

```
@Data
public class Orders {
    private Integer id;
    private Date datetime;
    private Double total;
    private Integer uid;
}
```

OrderMapper

```
public interface OrdersMapper {
    public List<Orders> findById(Integer id);
}
```

OrdersMapper.xml

```
<?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="test.mapper2.OrdersMapper">
    <!--namespace根据自己需要创建的的mapper的路径和名称填写-->


    <select id="findById" resultType="test.entity2.Orders">
        select * from orders where id = #{id}
    </select>
</mapper>
```

UserMapper

```
public interface UserMapper {
    public User findById(Integer id);
}
```

UserMapper.xml

```
<?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="test.mapper2.UserMapper">
    <!--namespace根据自己需要创建的的mapper的路径和名称填写-->
```

```
    <resultMap id="userMap" type="test.entity2.User">
        <id column="id" property="id"></id>
        <result column="username" property="username"></result>
        <result column="password" property="password"></result>
        <result column="birthday" property="birthday"></result>
        <association
                property="orders"
                javaType="test.entity2.Orders"
                column="uid"
                select="test.mapper2.OrdersMapper.findById"
                ></association>
    通过uid映射到表order，selct调用ordersMapper中的findById方法将查询到的值封装为Orders对象赋给UserMapper中property属性的orders

    </resultMap>

    <select id="findById" resultMap="userMap">
        select  * from user where id = #{id}
    </select>
</mapper>
```

config.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>

    <settings>
<!--        打印SQL-->
        <setting name="logImpl" value="STDOUT_LOGGING"/>
        <!--开启延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
    </settings>
    <environments default="dev">
        <environment id="dev">
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/test"/>
                <property name="username" value="root"/>
                <property name="password" value="123456"/>
            </dataSource>
        </environment>
    </environments>

    <mappers>
        <mapper resource="test/mapper/userMapper.xml"></mapper>
        <mapper resource="test/mapper2/UserMapper.xml"></mapper>
        <mapper resource="test/mapper2/OrdersMapper.xml"></mapper>
    </mappers>

</configuration>
```

```
public class test {
```

```
    public static void main(String[] args) {
        InputStream resourceAsStream =
test.class.getClassLoader().getResourceAsStream("config.xml");
        SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
        SqlSessionFactory sqlSessionFactory = builder.build(resourceAsStream);
        SqlSession sqlSession = sqlSessionFactory.openSession();
        userMapper mapper = sqlSession.getMapper(userMapper.class);
         UserMapper mapper1 = sqlSession.getMapper(UserMapper.class);
        User users = mapper1.findById(2);
        String name = users.getUsername();
        System.out.println(name);
        sqlSession.commit();



    }
}
```

```
D:\Java\jdk1.8.0_281\bin\java.exe ...
Logging initialized using 'class org.apache.ibatis.logging.stdout.StdOutImpl' adapter.
PooledDataSource forcefully closed/removed all connections.
PooledDataSource forcefully closed/removed all connections.
PooledDataSource forcefully closed/removed all connections.
PooledDataSource forcefully closed/removed all connections.
Opening JDBC Connection
Created connection 612097453.
Setting autocommit to false on JDBC Connection [com.mysql.cj.jdbc.ConnectionImpl@247bddad]
==>  Preparing: select * from user where id = ?
==> Parameters: 2(Integer)
<==    Columns: id, username, password, birthday
<==        Row: 2, lisi, 123, null
<==      Total: 1
lisi

进程已结束，退出代码 0
```

3.MyBatis缓存

缓存也是用来提高效率的一种方式，类似于字符串常量池、数据库连接池

缓存的作用是减少java程序和数据库的交互次数，从而提升程序运行效率，MyBatis查询出一个结果之后，将该结果存入缓存，下一次可以直接从缓存中取出结果，而不需要连接数据库进行查询

如果存入缓存之后做了修改操作，则MyBatis会清空缓存以保证数据的时效性

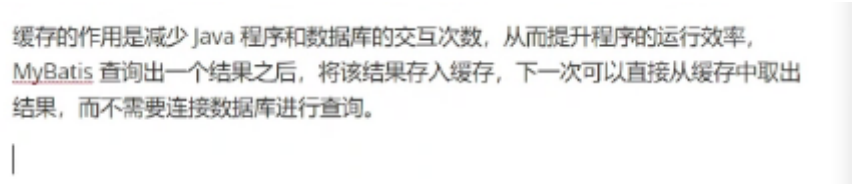
MyBatis缓存

1.一级缓存，默认开启且无法关闭的，同一个SqlSession中有效

```
StudentMapper mapper = sqlSession.getMapper(StudentMapper.class);
Student student = mapper.findById(15);
System.out.println(student.getName());
sqlSession.close();
sqlSession = factory.openSession();
mapper = sqlSession.getMapper(StudentMapper.class);
Student student1 = mapper.findById(15);
System.out.println(student1.getName());
```

2.二级缓存，比一级缓存作用域更大的一个缓存，Mapper级别的，只要同一个Mapper，无论多少个sqlsession，二级缓存都是有效的

二级缓存



耳机缓存默认是关闭的，需要手动开启，修改config.xml

```
<!-- 开启二级缓存-->
<setting name="cacheEnabled" value="true"/>
```

```
<?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>

    <settings>
<!--        打印SQL-->
        <setting name="logImpl" value="STDOUT_LOGGING"/>
        <!--开启延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
        <!-- 开启二级缓存-->
        <setting name="cacheEnabled" value="true"/>
    </settings>
    <environments default="dev">
        <environment id="dev">
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/test"/>
                <property name="username" value="root"/>
                <property name="password" value="123456"/>
            </dataSource>
        </environment>
    </environments>

    <mappers>
        <mapper resource="test/mapper/userMapper.xml"></mapper>
        <mapper resource="test/mapper2/UserMapper.xml"></mapper>
        <mapper resource="test/mapper2/OrdersMapper.xml"></mapper>
```

```
    </mappers>

</configuration>
```

Mapper.xml添加缓存标签

```
<?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="test.mapper2.UserMapper">
    <!--namespace根据自己需要创建的的mapper的路径和名称填写-->

    <cache></cache>
    <resultMap id="userMap" type="test.entity2.User">
        <id column="id" property="id"></id>
        <result column="username" property="username"></result>
        <result column="password" property="password"></result>
        <result column="birthday" property="birthday"></result>
        <association
                property="orders"
                javaType="test.entity2.Orders"
                column="uid"
                select="test.mapper2.OrdersMapper.findById"
                ></association>
    </resultMap>

    <select id="findById" resultMap="userMap">
        select  * from user where id = #{id}
    </select>
</mapper>
```

实体类实现序列化

```
import java.io.Serializable;
import java.util.List;

@Data
public class User  implements Serializable {
    private Integer id;
    private String username;
    private String password;
    private String birthday;
    private List<Orders> orders;
}
```

4.MyBatis动态SQL

根据参数信息动态生成SQL语句，不同的参数会创建不同的sql来完成数据查询操作

if判断当前属性是否为null，如果为null，statement就不会添加到SQL中，

where是判断and关键字是否会和where关键拼接到一起，如果拼接到一起则自动删除and关键字

OrederMapper

```
public interface OrdersMapper {
    public List<Orders> findById(Integer id);
    public Orders findAll(Orders orders);
}
```

```
<?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="test.mapper2.OrdersMapper">
    <!--namespace根据自己需要创建的的mapper的路径和名称填写-->


    <select id="findById" resultType="test.entity2.Orders">
        select * from orders

    </select>
    <select id="findAll" resultType="test.entity2.Orders">
        select  * from orders
        <where>
            <if test="id !=null">
                id=#{id}
            </if>
            <if test="datetime != null">
                and datetime = # {datetime}
            </if>
            <if test="total != null">
                and total = #{total}
            </if>
            <if test=" uid !=null" >
                and uid = #{uid}
            </if>
        </where>
    </select>
</mapper>
```

```
public class test {
    public static void main(String[] args) {
        InputStream resourceAsStream =
test.class.getClassLoader().getResourceAsStream("config.xml");
        SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
        SqlSessionFactory sqlSessionFactory = builder.build(resourceAsStream);
        SqlSession sqlSession = sqlSessionFactory.openSession();
        userMapper mapper = sqlSession.getMapper(userMapper.class);
        OrdersMapper ordersMapper = sqlSession.getMapper(OrdersMapper.class);
        Orders orders = new Orders();
        orders.setId(2);
        orders.setUid(1);
//        orders.setTotal((double) 5800);
        Orders all = ordersMapper.findAll(orders);
        System.out.println(all.getId());
        System.out.println(all.getTotal());
        System.out.println(all.getDatetime());
    }
}
```

choose、when 和if标签的作用相同

```
<select id="findAll" resultType="test.entity2.Orders">
    select  * from orders
      <where>
            <choose>
                <when test="id !=null">
                id=#{id}
                </when>
                <when test="datetime != null">
                    and datetime = # {datetime}
                </when>
                <when test="total != null">
                    and total = #{total}
                </when>
                <when test=" uid !=null" >
                    and uid = #{uid}
                </when>

            </choose>

        </where>
</select>
```

set

```
public interface OrdersMapper {
    public List<Orders> findById(Integer id);
    public Orders findAll(Orders orders);
    public void update(Orders orders);
}
```

```
<update id="update">
    update orders
    <set>
        <if test="total">
            total = #{total}
        </if>
        <if test="uid">
            uid = #{uid}
        </if>
    </set>
    where id =#{id}
</update>
```

```
 OrdersMapper ordersMapper = sqlSession.getMapper(OrdersMapper.class);
Orders orders = new Orders();
    orders.setId(2);
    orders.setTotal((double) 2000);
    ordersMapper.update(orders);
    sqlSession.commit();
```

foreach

将遍历的创建成一个集合

```
@Data
public class Orders {
    private Integer id;
    private Date datetime;
    private Double total;
    private Integer uid;
    private List<Integer> ids;
}
```

```
public interface OrdersMapper {
    public List<Orders> findById(Integer id);
    public Orders findAll(Orders orders);
    public void update(Orders orders);
    public List<Orders> findByIds(Orders orders);
}
```

```
<select id="findByIds" resultType="test.entity2.Orders">
        select * from orders
        <where>
            <foreach collection="ids" item="id" separator="," open="id in ("
close=")">
                #{id}
            </foreach>
        </where>

</select>
```

```
OrdersMapper ordersMapper = sqlSession.getMapper(OrdersMapper.class);
 Orders orders = new Orders();
 orders.setIds(Arrays.asList(1,2,3,5));
  List<Orders> byIds = ordersMapper.findByIds(orders);
   sqlSession.commit();
```

使用配置文件比较繁琐：

builder->factory->sqlsession->mapper

要使用MyBatis需要加载的组件很多，先创建很多对象，才能获取mapper，进行操作，开发步骤比较多

如果希望直接获取mapper进行操作

可以使用Spring框架帮助开发者自动生成所需要的各种组件，并完成依赖注入组装，开发者直接使用即可

Spring

IOC和AOP

IOC控制反转，让spring自动生成程序中所需要的各种组件，开发者直接用，不需要创建，开箱即用

AOP面向切面编程，面向对象的一种补充，做到核心业务和非核心业务的解耦合

IoC

解决的是项目中所需要组件的创建问题，不再由开发者创建项目中所需要的组件，，而是由Spring ioc

1、创建 Spring 工程

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.15</version>
</dependency>
```

2、在 spring.xml 中配置 bean，告诉 Spring 框架你需要的对象

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean class="com.southwind.entity.User" id="user">
        <property name="id" value="1"></property>
        <property name="name" value="张三"></property>
        <property name="age" value="22"></property>
    </bean>

</beans>
```

```
package com.southwind;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {
    public static void main(String[] args) {
        //启动Spring
        ApplicationContext applicationContext = new
ClassPathXmlApplicationContext("spring.xml");
        Object user = applicationContext.getBean("user");
        System.out.println(user);
    }
}
```