# Mybatis

环境搭建：

<?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>

<!--定义别名，类名作别名 -->

<typeAliases>

<package name=*"com.jun.mybatis.pojo"* />

</typeAliases>

<!--定义数据源 default引用environment的id，当前使用的环境 -->

<environments default=*"development"*>

<!--声明开发环境节点，可以配置多个节点，id切换 -->

<environment id=*"development"*>

<!-- 事务管理 ：使用原生jdbc事务-->

<transactionManager type=*"JDBC"* />

<!-- 数据源 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"url"*  value=*"jdbc:mysql://localhost:3306/mybatistest?chara cterEncoding=UTF-8"* />

<property name=*"username"* value=*"root"* />

<property name=*"password"* value=*"admin"* />

</dataSource>

</environment>

</environments>

<!--映射文件 -->

<mappers>

<mapper resource=*"com/jun/mybatis/pojo/teacherMapper.xml"*/>

<!-- <package name="com/jun/mybatis/mapper" /> -->

</mappers>

</configuration>

环境搭建详解：

1. 事务管理

<!-- 事务管理 ：使用原生jdbc事务管理方式-->

<transactionManager type=*"JDBC"* />

<!-- 把事务管理转交给其他容器：转交给sping管理 -->

<transactionManager type="MANAGED"></transactionManager>

1. 数据源

<!-- 数据源 dataSource：POOLED(使用数据库连接池)，-->

<dataSource type=*"POOLED"*>

**数据库连接池**：在web项目的META-INF中存放context.xml，在context.xml中编写数据库链接池相关的属性

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

<Context>

<Resource

driverClassName=*"com.mysql.jdbc.Driver"*

url=*"jdbc:mysql://localhost:3306/"*

username=*"root"*

password=*"admin"*

MaxActive=*"50" ：最大连接数*

MaxIdel=*"20" ：最大空闲数*

name=*"test" ：连接池名称*

auth=*"Container" ：*

maxWait=*"10000" ：最大等待时间 ms*

/>

</Context>

使用jndi获取数据库连接池中对象：

Context cxt = new InitialContext();

DataSource ds =(DataSource)cxt.lookup(*"java:comp/env/test"*);

Connection conn = ds.getConnection();

三种查询方式：

1. selectList() 返回值：List<resultType属性控制>

适用于查询结果都需要遍历的需求

**Xml:**

<select id=*"listTeachers"* resultType=*"Teacher"*>

select \* from teacher

</select>

**测试类：**

List<Teacher> teachers = session.selectList("listTeachers");

**for** (Teacher teacher : teachers) {

System.***out***.println(teacher);}

1. selectOne() 返回值：Object

适用于返回结果只是变量或一行数据时

**Xml:**

<!-- 查询总数量 -->

<select id="selTeacherCount" resultType="int">

select count(\*) from teacher

</select>

**测试类：**

**int** count = session.selectOne("selTeacherCount");

System.***out***.println(count);

1. selectMap() 返回值：Map()

适用于需要在查询结果中通过某列的值取到数据需求

**Xml:**

<select id=*"listTeachers2"* resultType=*"Teacher"*>

select \* from teacher

</select>

=================================================================

**测试类：**

Map<Object, Object> map = session.selectMap("listTeachers2", "teacher\_name");

System.***out***.println(map);

**Map<k,v>:k："teacher\_name" V由resultType=*"Teacher"控制***

### <settings>标签

1. 在mybatis全局配置文件中通过<settings>标签控制mybatis全局开关
2. 在mybatis.xml中开启log4j

必须要有log4j.jar,在src下有log4j.properties

<!--开启log4j日志 -->

<settings >

<setting name=*"logImpl"* value=*"LOG4J"*/>

</settings>

**log4j.properties**

//输出级别：fatal(致命错误)>erro(错误)>warn(警告)>info(普通信息)>debug

// log4j.rootCategory=INFO, CONSOLE，LOGFILE 显示在控制台且保存 //到LOGFILE文件

log4j.rootCategory=INFO, CONSOLE

//输出目的地

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout

//%C：输出类 %d{YYYY-MM-DD HH:mm:ss}:时间 %m：输出信息 %n：换行符

log4j.appender.CONSOLE.layout.ConversionPattern=- %m%n

log4j.appender.LOGFILE=org.apache.log4j.FileAppender

//文件地址

log4j.appender.LOGFILE.File=axis.log

log4j.appender.LOGFILE.Append=true

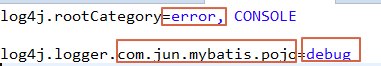
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout

log4j.appender.LOGFILE.layout.ConversionPattern=%C %m %n

1. log4j中可以输出指定内容的日志（控制某个局部内容的日志级别）

* 命名级别（包级别）：namespace的值

先设置总体级别调成Error不输出无用信息，再设置某个指定位置级别为debug



* 类级别：naspacespace后接类名
* 方法级别：naspacespace+类名+id属性值

### parameterType属性

1. sqlSession的selectList()和selectOne()的第二个参数，selectMap()的第三个参数都表示方法的参数

**parameterType**：参数类型

**#{}获取参数内容**：从0开始：#{0}，也可以：#{param1}表 示第一个参数

<select id=*"selById"* resultType=*"Teacher"* parameterType=*"Integer"*>

select \* from teacher where teacher\_id = **#{0}**

</select>

//2表示传入的id为2

Teacher teacher = session.selectOne("selById", 2);

如果**参数是map**:写成#{key}

<select id=*"selByIdName"* resultType=*"Teacher"* parameterType=*"map"*>

select \* from teacher where teacher\_id = #{**id**} and teacher\_name=#{**name**}

</select>

Map<String, Object> map = **new** HashMap();

map.put("**id**", 2);

map.put("**name**", "张三丰");

Teacher teacher = session.selectOne("selByIdName", map);

1. 分页查询

<!-- 分页查询 -->

<select id=*"selPage"* resultType=*"Teacher"* parameterType=*"map"*>

select \* from teacher limit #{pageStart},#{pageSize}

</select>

// 一页显示几个

**int** pageSize = 3;

// 第几页

**int** pageNumber = 1;

// 如果希望传递多个参数，可以使用对象或map

Map<String, Object> map = **new** HashMap();

map.put("pageSize", pageSize);

map.put("pageStart", pageSize \* (pageNumber - 1));

List<Teacher> teacher = session.selectList("selPage", map);

## Mybatis实现新增

1. 概念
   * 功能：从应用程序角度出发，软件具有哪些功能
   * 业务：完成功能是的逻辑
   * 事务：从数据库角度出发，完成业务时需要执行的SQL集合

<!--增加一个老师 -->

<insert id=*"addTeacher"* parameterType=*"String"*>

insert into teacher (teacher\_name) values(#{teacher\_name});

</insert>

**int** insert = session.insert("addTeacher", "zxm");

<!--增加一个老师 -->

<insert id=*"addTeacher2"* parameterType=*"Teacher"*>

insert into teacher values(default,#{teacher\_name});

</insert>

teacher.setTeacher\_name("zxm");

int insert = session.insert("addTeacher2", teacher);

## Mybatis接口绑定

实现步骤：

1. 接口包名和接口名与mapper.xml中<mapper>namespace相同
2. 接口中的方法名和mapper.xml标签的id属性相同
3. 在mybatis.xml中使用<package>进行接口扫描

**//接口中定义mapper中的id方法**

//1.1

List<Account> selAccIn2(String accIn, String accOut);

//2.1 @Param("accIn") 引号中的内容一定要和#{}中的内容相同

List<Account> selAccIn2(@Param("accIn")String accIn,@Param("accOut")String accOut);

**//xxxMapper.xml中com.jun.service.AccountService为接口全名**

<mapper namespace="com.jun.service.AccountService">

//1.1

</select>

<select id=*"selAccIn2"* resultType=*"Account"* >

select \* from account where accno=#{0} and name=#{1}

</select>

//2.1

<select id=*"selAccIn3"* resultType=*"Account"* >

select \* from account where accno=#{accIn} and name=#{accOut}

</select>

/**/mybatis-config.xml中 直接映射包名**

<mappers>

<package name="com/jun/mapper" />

</mappers>

## 动态sql

1. **If**标签：模糊查询，这时参数可以为任意组合。若都为null，则查询全部

**public** List<Log> selByAccInNoAccOutNo(@Param("accOutNo") String accOutNo, @Param("accInNo") String accInno);

<select id=*"selByAccInNoAccOutNo"* resultType=*"Log"*>

select \* from log where 1=1

<if test=*"accInNo!=null and accInNo !=''"*> **参数要和方法中的参数一致**

and accInNo=${accInNo}

</if>

<if test=*"accOutNo!=null and accOutNo!=''"*>

and accOutNo=${accOutNo}

</if>

</select>

1. **Where :**当编写where标签时，如果内容中第一个是**and,**则去掉第一个and，如果标签中有内容，则会**生成where关键字，**没有 就不生成

<select id=*"selByAccInNoAccOutNo2"* resultType=*"Log"*>

select \* from log

<where>

<if test=*"accInNo!=null and accInNo!=''"*>

and accInNo=${accInNo}

</if>

<if test=*"accOutNo!=null and accInNo!=''"*>

and accOutNo=${accOutNo}

</if>

</where>

</select>

**不传入参数时会sql异常**

只传入一个参数 ：select \* from log WHERE accInNo=3

同时传入两个参数：select \* from log WHERE accInNo=3 and accOutNo=1

1. **choose、when、otherwise: 只要有一个成立，其他都不执行**

<select id=*"selByAccInNoAccOutNo3"* resultType=*"Log"*>

select \* from log

<where>

<choose>

<when test=*"accOutNo!=null and accOutNo!=''"*>

and accOutNo=${accOutNo}

</when>

<when test=*"accInNo!=null and accInNo!=''"*>

and accInNo=${accInNo}

</when>

</choose>

</where>

</select>

当传入两个参数时，若第一个成立，则只执行第一个

select \* from log WHERE accOutNo=1

1. **set:**在修改sql中set从句 作用：去掉最后一个逗号，如果set里面有内容则生成set关键字，没有则不生成

<update id=*"update"* parameterType=*"Log"*>

update log

<set>

id=${id},<-- 防止不生成set关键字，根据sql语法要求需要“，” -->

<if test=*"accOutNo!=null and accOutNo!=''"*>

accInNo=${accInNo},

</if>

<if test=*"accInNo!=null and accInNo!=''"*>

accOutNo=${accOutNo},

</if>

</set>

where id=${id}

</update>

5）**trim：**其中有四个参数：

只输入id时：update log SET id=1 where id=1

输入id，且输入2个参数时：update log SET id=1, accInNo=3, accOutNo=1 where id=1

* + prefix=“a” ：在前面加 a
  + prefixOverrides=“a”：在前面去掉a
  + suffix=“a”:在后面加 a
  + suffixOverrides=“a” 在后面去掉 a

6）**bind：**重新设置值 ： <bind name=*"money"* value=*"'$'+money"*/>

输出时会输出：**$100.0**

7）**foreach：**循环参数内容，在内容前后添加内容，添加分隔符功能

适用于：in查询（select \* from log where id in(1,2,3) id=1，2,3的全部查询出来）

<select id=*"selIn"* parameterType=*"list"* resultType=*"Log"*>

select \* from log where id in

<foreach collection=*"list"* item=*"aaa"* open=*"("* close=*")"* separator=*","*>

#{aaa}

</foreach>

</select>

select \* from log where id in ( ? , ? )

item：迭代变量 open/ close：前面/后面 需要加的内容 separator：分隔符

## Mybatis多表联合查询

#### resultMap属性

1. resultMap实现单表映射

<resultMap type=*"teacher"* id=*"mymap"*>

<!--主键使用 id 标签配置映射关系 property:对应实体类中的属性 -->

<id column=*"t\_id"* property=*"t\_id"*/>

<!--其他列使用 result 标签配置映射关系 -->

<result column=*"t\_name"* property=*"t\_name"*/>

</resultMap>

<select id=*"selAll"* resultMap=*"mymap"*>

select \* from teacher

</select>

1. resultMap实现关联单个对象（N+1方式）

N+1查询方式：先查询出某个表的全部信息，根据这个标的信息查询另一个表的信息（全部由mybatis完成）

步骤：

* 在Student实体类中包含一个Teacher对象
* 在TeacherMapper中提供一个查询

<select id=*"selById"* resultType=*"Teacher"* parameterType=*"int"*>

select \* from teacher where id=#{0}

</select>

* 在studentMapper中:

< **association>：**关联一个对像时使用

Property:对象在类中的属性名

Select:通过那个查询查询出这个对象的信息

Column：把当前表的哪列值作为参数传给另一个查询

<resultMap type=*"Student"* id=*"stuMap"*>

<--<id column=*"s\_id"* property=*"s\_id"*/>

<result column=*"s\_name"* property=*"s\_name"*/>-->可省略

<result column=*"t\_id"* property=*"t\_id"*/>

<!-- 如果关联一个对象，使用 association pro-->

**<association property=*"teacher"* select=*"com.jun.mapper.TeacherMapper.selById"* column=*"tid"*>**

**</association>**

</resultMap>

<select id=*"selAll"* resultMap=*"stuMap"*>

select \* from student

</select>

1. resultMap实现关联多个对象（一对多）

<resultMap type=*"teacher"* id=*"mymap2"*>

<id column=*"t\_id"* property=*"t\_id"*/>

<result column=*"t\_name"* property=*"t\_name"*/>

<!--**collection：关联多个对象使用** ofType：泛型中的类型 -->

**<collection property=*"students"* ofType=*"Student"* select=*"com.jun.mapper.StudentMapper.selById"* column=*"t\_id"* >**

**</collection>**

</resultMap>

1. resultMap 实现联合查询

<resultMap type=*"Teacher"* id=*"mymap3"*>

<id column=*"t\_id"* property=*"t\_id"*/>

<result column=*"t\_name"* property=*"t\_name"*/>

**<collection property=*"students"* ofType=*"Student"*>**

**<id column=*"s\_id"* property=*"s\_id"*/>**

**<result column=*"s\_name"* property=*"s\_name"*/>**

**</collection>**

</resultMap>

<select id=*"selAll3"* resultMap=*"mymap3"*>

select t.t\_id,t\_name,s\_id,s\_name,s.t\_id

from teacher t **left join student s on t.t\_id=s.t\_id**

</select>

## 编写OpenSessionInView

**Util类编写**

**public** **class** MybatisUtil {

**private** **static** SqlSessionFactory *sqlSessionFactory* = **null**;

**private** **static** ThreadLocal<SqlSession> *tl* = **new** ThreadLocal<>();

**static** {

**try** {

InputStream in = Resources.*getResourceAsStream*("mybatis-config.xml");

*sqlSessionFactory* = **new** SqlSessionFactoryBuilder().build(in);

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **static** SqlSession getSqlSession() {

// 先从线程中获取session，如过没获取到 则开启session放入线程中

SqlSession session = *tl*.get();

**if** (session == **null**) {

session = *sqlSessionFactory*.openSession();

// 放入线程

*tl*.set(session);

}

**return** *tl*.get();

}

// 关闭session

**public** **static** **void** closeSession() {

SqlSession session = *tl*.get();

**if** (session != **null**) {

session.close();

}

*tl*.set(**null**);

}

}

**过滤器类编写**

@WebFilter("/\*")

**public** **class** OpenSessionInView **implements** Filter {

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

**throws** IOException, ServletException {

SqlSession session = MybatisUtil.*getSqlSession*();

**try** {

chain.doFilter(request, response);

} **catch** (Exception e) {

session.rollback();

e.printStackTrace();

} **finally** {

MybatisUtil.*closeSession*();

}

}

## Mybatis注解

作用：简化mapper.xml文件，但涉及动态sql的部分仍使用mapper.xml

@Select("select \* from teacher")

**public** List<Teacher> selAll4();

@Insert("insert into teacher values(default,#{t\_name})")

**public** **int** insTeacher(Teacher teacher);

@Delete("delete from teacher where t\_id=#{0}")

**public** **int** delTeacherById(**int** t\_id);