### MyBatis

#### 使用mybaits

##### 什么是MyBatis

.mybatis是一个持久层的轻量级orm框架，orm:object relative map 对象关系映射

mybatis是一个轻量级的jdbc框架

##### 1.2mybatis的缺点

1. 存在硬编码（java代码和sql代码分离）分页
2. 数据库管理不方便

3.数据库连接频繁关闭和打开（connection--mysql网络）

连接池 DataSource类，连接池的大小=2\*内核+1（大编发量）

4.preparedStatement预处理占位符处理麻烦

5.查询数据需要手动封装的对象，放入集合

##### 1.3.mybatis的优点

##### 1.4使用MyBatis步骤

###### 1.4.1导入包

mybatis-3.2.7.jar

asm-3.3.1.jar

cglib-2.2.2.jar

commons-logging-1.1.1.jar

javassist-3.17.1-GA.jar

log4j-1.2.17.jar

log4j-api-2.0-rc1.jar

log4j-core-2.0-rc1.jar

slf4j-api-1.7.5.jar

slf4j-log4j12-1.7.5.jar

###### 1.4.2.编写Mybatis配置文件

1.头部<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

可取API文档复制

2.,<configuration></configuration>

3.加载数据库信息

3.1.配置<properties></properties>（加载jdbc.properties数据库信息文件）

属性：这些属性都是可外部配置且可动态替换的，既可以在典型的 Java 属性文件中配置，亦可通过 properties 元素的子元素来传递

3.2<environments></environments>(配置环境)

尽管可以配置多个环境，每个 SqlSessionFactory 实例只能选择其一

即在测试的时候配置

3.3其中的<environment></environment>可以配置多个

注意这里的关键点:

默认的环境 ID（比如:default=”development”）。

每个 environment 元素定义的环境 ID（比如:id=”development”）。

事务管理器的配置（比如:type=”JDBC”）。

数据源的配置（比如:type=”POOLED”）。

默认的环境和环境 ID 是自我解释的。你可以使用你喜欢的名称来命名，只要确定默认的要匹配其中之一。

事务管理器（transactionManager

3.4事务管理器（transactionManager）

3.5数据源（dataSource）

4.sql映射文件（四种方式）

4.1<mappers></mappers>

4.2

1.<mappers>

<mapper resource="org/mybatis/builder/AuthorMapper.xml"/>

<mapper resource="org/mybatis/builder/BlogMapper.xml"/>

<mapper resource="org/mybatis/builder/PostMapper.xml"/>

</mappers>

4.3

2.<!-- Using url fully qualified paths -->

<mappers>

<mapper url="file:///var/mappers/AuthorMapper.xml"/>

<mapper url="file:///var/mappers/BlogMapper.xml"/>

<mapper url="file:///var/mappers/PostMapper.xml"/>

</mappers>

4.4

3.<!-- Using mapper interface classes -->

<mappers>

<mapper class="org.mybatis.builder.AuthorMapper"/>

<mapper class="org.mybatis.builder.BlogMapper"/>

<mapper class="org.mybatis.builder.PostMapper"/>

</mappers

4.5

4.<!-- Register all interfaces in a package as mappers -->

<mappers>

<package name="org.mybatis.builder"/>

</mappers>

4.6.示例<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 一、加载数据库信息 -->

<!-- 加载jdbc.properties数据库信息文件 -->

<properties resource="jdbc.properties">

</properties>

<environments default="development">

<!-- 对应的数据库信息 -->

<environment id="development">

<!-- 事务管理方式：采用jdbc -->

<transactionManager type="JDBC" />

<!-- 数据源的类型 -->

<dataSource type="POOLED">

<!-- 驱动 -->

<property name="driver" value="${driver}" />

<!-- 协议地址 -->

<property name="url" value="${url}" />

<!-- 用户名 -->

<property name="username" value="${username}" />

<!-- 密码 -->

<property name="password" value="${password}" />

</dataSource>

</environment>

</environments>

<!-- 二、sql映射文件-->

<mappers>

<mapper resource="sqlmap/User.xml"/>

</mappers>

</configuration>

第一个resource是隐式单个，package是映射一个包下的所有sqlxml文件

###### 1.4.3.配置sql映射文件

1.文件命名：实体类名.xml（早起） 实体类名+Mapper.xml

2.头部

<?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">

可到官方API文档找到复制

1. Mapper

3.1

mapper:sql映射文件的根元素

namespace:命名空间，隔离不同的sql映射（隔离不同实体类的sql）

3.2查询语句

3.2.1 示例

<select id="findAllUser" resultType="com.cc.entity.User">

select id, username,birthday,sex,address from user

</select>

3.2.2

id就是MappedStatement的唯一标识

jdbc中可以构建statement对象，preparedStatement对象

statement对象(父接口)

| ======>preparedStatement对象(子接口)

|=====>MappedStatement对象

mybatis中是创建MappedStatement对象

3.2.3

resultType：查询的结果集封装成什么类的对象，同时还可以放入集合

3.3

3.3.1示例

<select id="findById" parameterType="int" resultType="com.cc.entity.User">

select id, username,birthday,sex,address from user where id =#{id}

</select>

3.3.2

parameterType：填写参数的类型

#{}站位符同时绑定参数，如果是基本类型可以随意

为什么不写$， ${} 取值以后再去编译SQL语句

${} 是未经过预编译的,仅仅是取变量的值

3.3模糊查询

3.3.1示例

<select id="findByName" parameterType="java.lang.String" resultType="com.cc.entity.User">

select id, username,birthday,sex,address from user where username like '%${value}%'

</select>

3.3.3

#{} 编译好 SQL 语句再取值

# 将传入的数据都当成一个字符串，会对自动传入的数据加一个单引号。

${} 取值以后再去编译SQL语句

${} 是未经过预编译的,仅仅是取变量的值

3.4添加

3.4.1示例

1.示例

<insert id="addUser" parameterType="com.cc.entity.User" >

<!-- insert into user values();必须要添加所有字段的值 -->

<!-- insert into user(username,birthday) values('二狗','1900-2-2') 必须带传2个值 -->

insert into user

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="username !=null">

username,

</if>

<if test="birthday!=null">

birthday,

</if>

<if test="sex!=null">

sex,

</if>

<if test="address!=null">

address,

</if>

</trim>

values

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="username !=null">

#{username},

</if>

<if test="birthday!=null">

#{birthday},

</if>

<if test="sex!=null">

#{sex},

</if>

<if test="address!=null">

#{address},

</if>

</trim>

</insert>

##### 1.1

##### 物理分页和内存分页

###### 1.1.1物理分页

1. 先去https://github.com/pagehelper/Mybatis-PageHelper下载分页插件jar包，和，然后导入（此处2个jar包必须对应，所以都下载最新的）
2. select分页语句  
    select \* from employee limit 0,3;
3. 分页插件

mybatis-pageHelper（讲解）

https://github.com/pagehelper/Mybatis-PageHelper

mybatis-plus （自学）

###### 1.1.2内存分页

先将查询的数据加载内存RowBound，然后再在内存中进行分页

##### 1.2怎么使用mybatis-pageHelper插件？

###### 下载并导入包（最新版本）

http://repo1.maven.org/maven2/com/github/pagehelper/pagehelper/

http://repo1.maven.org/maven2/com/github/jsqlparser/jsqlparser/

###### mybatis-config.xml中配置

<!-- 分页插件 -->

<plugins>

<plugin interceptor="com.github.pagehelper.PageInterceptor">

<!-- config params as the following -->

<!-- 数据库方言：mybatis根据方言生产符合对应数据库的sql语句 -->

<property name="properties" value="dialect=mysql" />

</plugin>

</plugins>

###### 使用

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

//紧跟着的第一个select方法会被分页

List<Country> list = countryMapper.selectIf(1);

assertEquals(2, list.get(0).getId());

assertEquals(10, list.size());

//分页时，实际返回的结果list类型是Page<E>，如果想取出分页信息，需要强制转换为Page<E>

assertEquals(182, ((Page) list).getTotal());

###### 示例

1. mybatis.xml配置

<plugins>

<plugin interceptor=*"com.github.pagehelper.PageInterceptor"*>

<property name=*"properties"* value=*"dialect=mysql"*/>

</plugin>

</plugins>

1. 接口中定义

@Select("select \* from user where id>1")

List<User>findPageSize(**int** i);

使用了注解

1. 测试类

@Test

**public** **void** testPageHelper() {

SqlSession sqlSession = sqlSessionFactory.openSession();

UserMapper mapper = sqlSession.getMapper(UserMapper.**class**);

PageHelper.*startPage*(2, 3);

List<User> user = mapper.findPageSize(1);

System.*out*.println(user);

// 输出所有的分页信息

System.*out*.println

((Page) user);

sqlSession.close();

}

##### 1.3注解的方式开发mapper

###### 定义mapper接口

public interface EmployeeMapper {

@Select("select \* from tbl\_employee where id=#{id}")

Employee findEmpById(Integer eid);

}

保障：mybatis-config.xml中通过包扫描mapper

###### 2测试

@Test

**public** **void** testPageHelper() {

SqlSession sqlSession = sqlSessionFactory.openSession();

UserMapper mapper = sqlSession.getMapper(UserMapper.**class**);

PageHelper.*startPage*(2, 3);

List<User> user = mapper.findPageSize(1);

System.*out*.println(user);

// 输出所有的分页信息

System.*out*.println

((Page) user);

sqlSession.close();

}