myBatis

1. 导入相关的jar包
2. 提供相应的配置文件，这里取名为mybatis-config.xml相当于Hibernate中的hibernate-config.xml文件，关于配置文件可参看文档，要注意的是

<typeAliases>

<typeAlias type=*"com.wangke.po.User"* alias=*"User"*/>

</typeAliases>

是给类型取别名

1. 创建UserMapper接口，mybatis值需要接口，而不像ibatis还需要实现类。

相当于dao的接口一样提供增删改查的方法。

1. 提供UserMapper的映射文件UserMapper.xml

<mapper namespace=*"com.wangke.mapper.UserMapper"*>

注意这里的namespace一定要是*UserMapper的全名*

<select id=*"findUserById"*

parameterType=*"java.lang.Integer"*

resultType=*"User"*>

select \* from t\_user where id = #{id}

</select>

<insert id=*"saveUser"* parameterType=*"User"*>

insert into t\_user values(null,#{username},#{age})

</insert>

</mapper>

1. 提供SqlSessionFactory的工具类

**private** **static** SqlSessionFactory *sqlSessionFactory*;

**static**{

String resource = "mybatis-config.xml";

Reader reader = **null**;

**try** {

reader = Resources.*getResourceAsReader*(resource);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

*sqlSessionFactory* = **new**

SqlSessionFactoryBuilder().build(reader);

}

**public** **static** SqlSessionFactory getSqlSessionFactory(){

**return** *sqlSessionFactory*;

}

1. 测试

SqlSessionFactory factory =

MyBatisUtil.*getSqlSessionFactory*();

SqlSession session = factory.openSession();

**try** {

UserDao userDao = session.getMapper(UserDao.**class**);

User user = **new** User();

user.setUsername("jack");

user.setAge(19);

userDao.saveUser(user);

session.commit();

} **catch** (Exception e) {

e.printStackTrace();

session.rollback();

}**finally**{

session.commit();

}

<http://wanqiufeng.blog.51cto.com/409430/515132>

1，本文采用的Mybatis框架版本为： mybatis-3.0.4.jar

2， 以持久化一个简单的User bean为例

User实体类的代码如下：

**package** bean;

**public** **class** User {

**private** String name;

**private** Integer age;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getAge() {

**return** age;

}

**public** **void** setAge(Integer age) {

**this**.age = age;

}

**public** User(String name, Integer age) {

**super**();

**this**.name = name;

**this**.age = age;

}

**public** User() {

**super**();

} //必须要有这个无参构造方法，不然根据UserMapper.xml中的配置，在查询数据库时，将不能反射构造出User实例

}

User实体的映射器取名为UserMapper(接口)，其代码如下

**package** Mapper;

**import** bean.User;

**public** **interface** UserMapper {

**public** **void** insertUser(User user);

**public** User getUser(String name);

}

注意，该接口类似于Hibernate中的DAO层接口，唯一不同的是，MyBatis中，只声明该接口就可以，不需要去实现.

UserMapper映射器对应的配置文件如下(相当于Hibernate中的User.hbm.xml,当然这里面不是属性映射字段，而是直接是映射器中各种方法对应的sql语句)：

<?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="Mapper.UserMapper">

<!-- 这里namespace必须是UserMapper接口的路径，不然要运行的时候要报错 “is *not* known *to* the *MapperRegistry*”-->

<insert *id*="insertUser" parameterType="User">

insert into vincent\_user(name,age) values(#{name},#{age})

<!-- 这里sql结尾不能加分号，否则报“ORA-00911”的错误 -->

</insert>

<!-- 这里的id必须和UserMapper接口中的接口方法名相同，不然运行的时候也要报错 -->

<select id="getUser" resultType="User" parameterType="java.lang.String">

select \* from vincent\_user where name=#{name}

</select>

</mapper>

MyBatis的配置文件这里取名为mybatis-config.xml其配置代码如下(相当与Hibernate的hibernate.cfg.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>

<!-- changes from the defaults **for** testing -->

<setting name="cacheEnabled" value="false" />

<setting name="useGeneratedKeys" value="true" />

<setting name="defaultExecutorType" value="REUSE" />

</settings>

<typeAliases>

<typeAlias alias="User" type="bean.User"/>

</typeAliases>

<environments **default**="development">

<environment id="development">

<transactionManager type="jdbc"/>

<dataSource type="POOLED">

<property name="driver" value="oracle.jdbc.driver.OracleDriver"/>

<property name="url" value="jdbc:oracle:thin:@192.168.56.201:1521:system"/>

<property name="username" value="vincent"/>

<property name="password" value="1234"/>

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="resource/UserMapper.xml" />

</mappers>

</configuration>

加载配置文件得到sqlSessionFactory的工具类

**package** Util;

**import** java.io.IOException;

**import** java.io.Reader;

**import** org.apache.ibatis.io.Resources;

**import** org.apache.ibatis.session.SqlSessionFactory;

**import** org.apache.ibatis.session.SqlSessionFactoryBuilder;

**public** **class** MyBatisUtil {

**private** **final** **static** SqlSessionFactory *sqlSessionFactory*;

**static** {

String resource = "resource/mybatis-config.xml";

Reader reader = **null**;

**try** {

reader = Resources.*getResourceAsReader*(resource);

} **catch** (IOException e) {

System.*out*.println(e.getMessage());

}

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

}

**public** **static** SqlSessionFactory getSqlSessionFactory() {

**return** *sqlSessionFactory*;

}

}

测试代码如下：

**package** TestMapper;

**import** org.apache.ibatis.session.SqlSession;

**import** org.apache.ibatis.session.SqlSessionFactory;

**import** org.junit.Test;

**public** **class** TestMapper {

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

**static** {

*sqlSessionFactory* = MyBatisUtil.*getSqlSessionFactory*();

}

@Test

**public** **void** testAdd() {

SqlSession sqlSession = *sqlSessionFactory*.openSession();

**try** {

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

User user = **new** User("tom",**new** Integer(5));

userMapper.insertUser(user);

sqlSession.commit();//这里一定要提交，不然数据进不去数据库中

} **finally** {

sqlSession.close();

}

}

@Test

**public** **void** getUser() {

SqlSession sqlSession = *sqlSessionFactory*.openSession();

**try** {

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

User user = userMapper.getUser("jun");

System.*out*.println("name: "+user.getName()+"|age: "+user.getAge());

} **finally** {

sqlSession.close();

}

}

}