# Mybatis 的概述

Mybatis 是一个使用java编写的持久层框架。它封装了 JDBC ，使开发者只需要关注 sql 语句，而无需关注注册驱动、创建连接、创建 Statement 等繁杂的过程。

# 数据准备

-- 创建数据库

CREATE DATABASE IF NOT EXISTS db\_mybatis CHARACTER SET utf8;

-- 创建数据表

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (

`id` int(11) NOT NULL auto\_increment,

`username` varchar(32) NOT NULL COMMENT '用户名称',

`birthday` datetime default NULL COMMENT '生日',

`sex` char(1) default NULL COMMENT '性别',

`address` varchar(256) default NULL COMMENT '地址',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 添加记录

insert into `user`(`id`,`username`,`birthday`,`sex`,`address`) values (41,'老王','2018-02-27 17:47:08','男','北京'),(42,'小二王','2018-03-02 15:09:37','女','北京金燕龙'),(43,'小二王','2018-03-04 11:34:34','女','北京金燕龙'),(45,'传智播客','2018-03-04 12:04:06','男','北京金燕龙'),(46,'老王','2018-03-07 17:37:26','男','北京'),(48,'小马宝莉','2018-03-08 11:44:00','女','北京修正');

# Mybatis 的环境搭建

---使用intellij idea新建maven项目：new>project>maven>next>....

可以去mybatis官网获取pom.xml的配置

在pom.xml 文件中导入依赖（一般只要mybatis和mysql）：

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.item</groupId>  
 <artifactId>day01\_chq\_mybatis</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <dependencies>  
 <!-- mybatis -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.4.5</version>  
 </dependency>  
 <!-- 数据库驱动 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.8</version>  
 <scope>runtime</scope>  
 </dependency>  
 <!-- 日志 -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
 <!-- JUnit -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
</project>

创建实体类,com.item.domain.User

package com.item.domain;  
  
import java.io.Serializable;  
import java.util.Date;  
  
public class User implements Serializable {  
 private Integer id;  
 private String username;  
 private Date birthday;  
 private String sex;  
 private String address;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public Date getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(Date birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getSex() {  
 return sex;  
 }  
  
 public void setSex(String sex) {  
 this.sex = sex;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "id=" + id +  
 ", username='" + username + '\'' +  
 ", birthday=" + birthday +  
 ", sex='" + sex + '\'' +  
 ", address='" + address + '\'' +  
 '}';  
 }  
}

创建实体类的持久层接口（mapper 或 dao）,com.item.dao.IUserMapper

package com.item.mapper;  
  
import com.item.domain.User;  
  
import java.util.List;  
  
public interface IUserMapper {  
 List<User> findAll();  
}

在resources下新建主配置文件xxx.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>  
 <!-- 全局变量 -->  
 <properties>  
 <property name="driver" value="com.mysql.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3306/db\_mybatis"/>  
 <property name="username" value="root"/>  
 <property name="password" value=""/>  
 </properties>  
  
 <!--配置环境-->  
 <environments default="development">  
 <environment id="development">  
 <!-- 配置事务类型 -->  
 <transactionManager type="JDBC"></transactionManager>  
 <!-- 配置数据源（连接池） -->  
 <dataSource type="POOLED">  
 <property name="driver" value="${driver}"/>  
 <property name="url" value="${url}"/>  
 <property name="username" value="${username}"/>  
 <property name="password" value="${password}"/>  
 </dataSource>  
 </environment>  
 </environments>  
  
 <!-- 指定映射文件 -->  
 <mappers>  
 <!-- 1.使用映射文件方式 -->  
 <mapper resource="mapper/UserMapper.xml"/>  
 <!-- 2.使用注解方式 -->  
 <!--<mapper class="com.item.mapper.IUserMapper"/>-->

</mappers>  
</configuration>

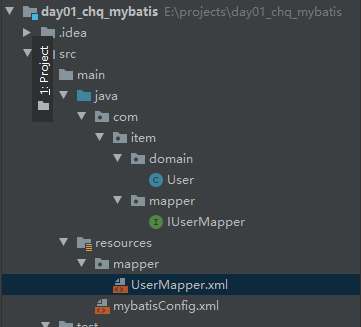
方式1：在resources/mapper下编写接口映射文件（UserMapper.xml）

<?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="com.item.mapper.IUserMapper">  
 <!-- 配置查询所有用户 -->  
 <select id="findAll" resultType="com.item.domain.User">  
 SELECT \* FROM user  
 </select>  
</mapper>

方式2：在IUserMapper添加注解；

public interface IUserMapper {  
 @Select("SELECT \* FROM user")  
 List<User> findAll();  
}

最终目录：



注意事项

在 Mybatis 中，持久层的操作接口名称和映射文件也叫 Mapper ，所以 UserMapper 和 IUserDao 是一样的

映射配置文件的 mapper 标签 namespace 属性的取值必须是 mapper 接口的全限定类名

映射配置文件的操作配置，id 属性的取值必须是 mapper 接口的方法名

在test.java下编写测试类com.item.test.MybatisTest 如下:

package com.item.test;  
  
import com.item.domain.User;  
import com.item.mapper.IUserMapper;  
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.InputStream;  
import java.util.List;  
  
public class MybatisTest {  
 public static void main(String[] args) throws Exception {  
 // 1. 读取配置文件  
 InputStream is = Resources.*getResourceAsStream*("mybatisConfig.xml");  
 // 2. 创建 SqlSessionFactory 工厂  
 SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();  
 SqlSessionFactory factory = builder.build(is);  
 // 3. 获取 SqlSession 对象  
 SqlSession sqlSession = factory.openSession();  
 // 4. 使用 SqlSession 创建 Mapper 的代理对象  
 IUserMapper mapper = sqlSession.getMapper(IUserMapper.class);  
 // 5. 使用代理对象执行查询  
 List<User> users = mapper.findAll();  
 for (User user:users){  
 System.*out*.println(user);  
 }  
 // 6. 释放资源  
 sqlSession.close();  
 is.close();  
 }  
  
}

案例分析：

