MyBatis高级编程

# 一、原始的JDBC编程

## JDBC编程（Junit测试）

|  |
| --- |
| public class JdbcConn3\_junit {  @Test  public void testJDBC() {  // 1、定义连接Connection对象  Connection conn = null;  // 2、定义预处理对象  PreparedStatement pstm = null;  // 3、定义结果集  ResultSet rs = null;  // 4、定义mysql驱动  String driverClassName = "com.mysql.jdbc.Driver";  // 5、定义数据库连接url  String url = "jdbc:mysql://localhost:3306/test?characterEncoding=utf-8";  // 6、定义数据库名  String userName = "root";  // 7、定义数据库密码  String password = "111";  try {  // 8、加载驱动  Class.forName(driverClassName);  // 9、获取连接  conn = DriverManager.getConnection(url, userName, password);  System.out.println(conn);  // 10、定义sql  String sql = "SELECT \* FROM user WHERE user\_name=?";  // 11、开启事务  conn.setAutoCommit(false);  // 12、创建sql预处理对象  pstm = conn.prepareStatement(sql);  // 13、设置参数  pstm.setString(1, "张三");  // 14、预处理对象操作sql查询数据库,返回结果集对象集合。  rs = pstm.executeQuery();  // 15、遍历结果集对象集合  while (rs.next()) {  System.err.println(rs.getString("user\_id") + "" + rs.getString("user\_name"));  }  // 16、 提交事务  conn.commit();  } catch (ClassNotFoundException e) {  e.printStackTrace();  } catch (SQLException e) {  e.printStackTrace();  try {  // 17、 回滚事务  conn.rollback();  } catch (SQLException e1) {  e1.printStackTrace();  }  } finally {  try {  // 18、关闭资源  if (rs != null) {  rs.close();  }  if (pstm != null) {  pstm.close();  }  if (conn != null) {  conn.close();  }  } catch (SQLException e) {  e.printStackTrace();  }  }  }  } |

# 二、MyBatis框架

## 2.1、MyBatis介绍

MyBatis的前身就是iBatis,本是apache的一个开源项目，2010年这个项目由apahce software foundation 迁移到了google code，并且改名为MyBatis。

**MyBatis是一个数据持久层(ORM)框架。**

MyBatis是把实体类和sql语句之间建立了映射关系，而Hibernate在实体类和数据库之间建立了映射关系。

**总体来说 MyBatis 主要完成两件事情**

1. 根据 JDBC 规范建立与数据库的连接；
2. 通过Annotaion/XML+JAVA反射技术，实现 Java 对象与关系数据库之间相互转化。

## 2.2、MyBatis优点

* 基于SQL语法，简单易学，适合于初学者。
* 能了解底层组装过程。
* SQL语句封装在配置文件中，便于统一管理与维护，降低了程序的耦合度。
* 程序调试方便。

  

## 2.3、类层次结构

|  |
| --- |
|  |

## 2.4、项目搭建

### 2.4.1 pom.xml文件中新增mybatis所需依赖包

**<!-- mybatis所需包 -->**

**<dependency>**

**<groupId>org.mybatis</groupId>**

**<artifactId>mybatis</artifactId>**

**<version>3.2.8</version>**

**</dependency>**

**<dependency>**

**<groupId>org.mybatis</groupId>**

**<artifactId>mybatis-spring</artifactId>**

**<version>1.3.0</version>**

**</dependency>**

<!-- Spring创建数据源的依赖包-->

<dependency>

<groupId>**org.springframework**</groupId>

<artifactId>**spring-jdbc**</artifactId>

<version>**4.2.5.RELEASE**</version>

</dependency>

### 2.4.2 applicationContext.xml文件中配置数据源

**<!-- c3p0数据源 -->**

**<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>**

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

**<property name=*"jdbcUrl"***

**value=*"jdbc:mysql://localhost:3306/test?characterEncoding=utf-8"*></property>**

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

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

**</bean>**

### 2.4.3 访问数据库的接口UserMapper

**public interface UserMapper {**

**//查询用户**

**public User getLoginUser(User user);**

**//注册用户**

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

**}**

### 2.4.4 建立映射文件userMapper.xml

注意：映射文件 中id的名称需要和userMapper文件中的方法名相同

**<?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.huadi.mapper.UserMapper"*>**

**<select id=*"getLoginUser"* parameterType=*"User"* resultType=*"User"*>**

**select \* from t\_account where username= #{username} and password = #{password}**

**</select>**

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

**insert into t\_account (username,password) values (#{username},#{password})**

**</insert>**

**</mapper>**

### 2.4.5 新建UserLoginService类

**public** **interface** UserLoginService {

**public** User getLoginUser(User user); //用户登录

**public** **void** registerUser(User user); //用户注册

}

### 2.4.6 新建UserLoginServiceImpl

**public** **class** UserLoginServiceImpl **implements** UserLoginService {

**注入Mapper**

@Autowired

**private** UserMapper userMapper;

**public** User getLoginUser(User user) {

**return** userMapper.getLoginUser(user);

}

**public** **void** registerUser(User user) {

userMapper.registerUser(user);

}

}

### 2.4.7 建立mybatis-config.xml全局配置文件

**说明：**

**mybatis-config.xml是mybatis的全局管理文件,可以实现对所有的映射文件和实体bean的配置和管理。**

**<?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>**

**<!-- 实体类,简称 -设置别名 -->**

**<typeAliases>**

**<typeAlias alias=*"User"* type=*"com.huadi.pojo.User"* />**

**</typeAliases>**

**<!-- 实体接口映射文件 -->**

**<mappers>**

**<mapper resource="mybatis/userMapper.xml" />**

**</mappers>**

**</configuration>**

### 2.4.8 配置sqlSessionFactory和MapperScannerConfigurer

**说明：sqlSessionFactory主要完成对数据库事务的管理，而MapperScannerConfigurer主要建立mapper接口和sql脚本之间的映射。**

***<!-- mybatis的SqlSession的工厂 -->***

**<bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>**

**<property name=*"dataSource"* ref=*"dataSource"*></property>**

**<property name=*"configLocation"* value=*"classpath:mybatis/mybatis-config.xml"* />**

**</bean>**

***<!-- mapper扫描器 -->***

**<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>**

**<property name=*"basePackage"* value=*"com.huadi.mapper"*></property>**

**<property name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"*></property>**

**</bean>**

### 2.4.9 applicationContext.xml配置用户业务类userLoginService

***<!-- 配置Service类 -->***

**<bean id=*"userLoginService"* class=*"com.huadi.service.impl.UserLoginServiceImpl"*></bean>**

### 2.4.10 通过Junit进行验证

