Mybatis

# 课程目标

Mybatis介绍

Mybatis增删改查

SQL注入问题介绍

Mybatis xml与注解实现

Mybatis分页

# Mybatis快速入门

## 2.1 Mybatis介绍

MyBatis是支持***普通SQL查询***，***存储过程***和***高级映射***的优秀持久层框架。MyBatis消除了几乎所有的JDBC代码和参数的手工设置以及对结果集的检索封装。MyBatis可以使用简单的XML或注解用于配置和原始映射，将接口和Java的POJO（Plain Old Java Objects，普通的Java对象）映射成数库中的记录.**JDBC-** 🡪**MyBatis-**🡪**Hibernate**

## 2.2 Mybatis环境搭建

### 2.2.1添加Maven坐标

|  |
| --- |
| **<dependencies>**  **<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->**  **<dependency>**  **<groupId>org.mybatis</groupId>**  **<artifactId>mybatis</artifactId>**  **<version>3.4.4</version>**  **</dependency>**  **<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->**  **<dependency>**  **<groupId>mysql</groupId>**  **<artifactId>mysql-connector-java</artifactId>**  **<version>5.1.21</version>**  **</dependency>**  **</dependencies>** |

### 2.2.2建表

|  |
| --- |
| CREATE TABLE users(id INT PRIMARY KEY AUTO\_INCREMENT, NAME VARCHAR(20), age INT);  INSERT INTO users(NAME, age) VALUES('Tom', 12);  INSERT INTO users(NAME, age) VALUES('Jack', 11); |

### 2.2.3添加mybatis配置文件

|  |
| --- |
| **<?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>**  **<environments default=*"development"*>**  **<environment id=*"development"*>**  **<transactionManager type=*"JDBC"* />**  **<dataSource type=*"POOLED"*>**  **<property name=*"driver"* value=*"com.mysql.jdbc.Driver"* />**  **<property name=*"url"* value=*"jdbc:mysql://localhost:3306/test"* />**  **<property name=*"username"* value=*"root"* />**  **<property name=*"password"* value=*"root"* />**  **</dataSource>**  **</environment>**  **</environments>**  **</configuration>** |

### 2.2.4定义表的实体类

|  |
| --- |
| **package com.entity;**  **public class User {**  **private int id;**  **private String name;**  **private int age;**  **//get,set方法**  **}** |

### 2.2.5定义userMapper接口

|  |
| --- |
| **package** com.itmayiedu.mapper;  **import** com.itmayiedu.entity.User;  **public** **interface** UserMapper {  **public** User getUser(**int** id);  } |

### 2.2.6定义操作users表的sql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.itmayiedu.mapper.UserMapper"*>  <select id=*"getUser"* parameterType=*"int"* resultType=*"com.itmayiedu.entity.User"*>  SELECT \*  FROM users where id =#{id}  </select>  </mapper> |

### 2.2.7mybatis.xml文件中加载配置文件

|  |
| --- |
| <mappers>  <mapper resource=*"mapper/userMapper.xml"* />  </mappers> |

### 2.2.8mybatis.xml测试方法

|  |
| --- |
| **import java.io.File;**  **import java.io.IOException;**  **import java.io.Reader;**  **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 com.itmayiedu.entity.User;**  **public class TestMybatis {**  **public static void main(String[] args) throws IOException {**  **String resource = "mybatis.xml";**  **// 读取配置文件**  **Reader reader = Resources.*getResourceAsReader*(resource);**  **// 获取会话工厂**  **SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);**  **SqlSession openSession = sqlSessionFactory.openSession();**  **// 查询**  **String sql = "com.itmayiedu.mapper.UserMapper.getUser";**  **// 调用api查询**  **User user = openSession.selectOne(sql, 1);**  **System.*out*.println(user.toString());**  **}**  **}** |

## 2.2.3增加案例

Xml:

|  |
| --- |
| **<insert id=*"addUser"* parameterType=*"com.itmayiedu.entity.User"* >**    **INSERT INTO users(NAME, age) VALUES(#{name}, #{age});**  **</insert>** |

代码:

|  |
| --- |
| **static public void add() throws IOException{**  **String resource = "mybatis.xml";**  **// 读取配置文件**  **Reader reader = Resources.*getResourceAsReader*(resource);**  **// 获取会话工厂**  **SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);**  **SqlSession openSession = sqlSessionFactory.openSession();**  **// 查询**  **String sql = "com.itmayiedu.mapper.UserMapper.addUser";**  **// 调用api查询**  **User userPa = new User();**  **userPa.setAge(19);**  **userPa.setName("张三");**  **int reuslt = openSession.insert(sql, userPa);**  **System.*out*.println(reuslt);**  **}** |

## 2.2.4 删除

Xml:

|  |
| --- |
| <delete id=*"delUser"* parameterType=*"int"* >  delete from users where id=#{id}  </delete> |

代码:

|  |
| --- |
| **static public void delUser() throws IOException{**  **String resource = "mybatis.xml";**  **// 读取配置文件**  **Reader reader = Resources.*getResourceAsReader*(resource);**  **// 获取会话工厂**  **SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);**  **SqlSession openSession = sqlSessionFactory.openSession();**  **// 查询**  **String sql = "com.itmayiedu.mapper.UserMapper.delUser";**  **int reuslt = openSession.delete(sql,1);**  **System.*out*.println(reuslt);**  **}** |

# sql注入案例

## 3.1创建表+测试数据

|  |
| --- |
| create table user\_table(  id int Primary key,  username varchar(30),  password varchar(30)  );  insert into user\_table values(1,'yushengjun-1','12345');  insert into user\_table values(2,'yushengjun-2','12345'); |

## 3.2 jdbc进行加载

|  |
| --- |
| **String username = "yushengjun-1";**  **String password = "12345";**  **String sql = "SELECT id,username FROM user\_table WHERE " + "username='" + username + "'AND " + "password='"**  **+ password + "'";**  **Class.*forName*("com.mysql.jdbc.Driver");**  **Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/test", "root", "root");**  **PreparedStatement stat = con.prepareStatement(sql);**  **System.*out*.println(stat.toString());**  **ResultSet rs = stat.executeQuery();**  **while (rs.next()) {**  **String id = rs.getString(1);**  **String name = rs.getString(2);**  **System.*out*.println("id:" + id + "---name:" + name);**  **}** |

## 3.3将username的值设置为

username='  OR 1=1 --

因为--表示SQL注释，因此后面语句忽略；

因为1=1恒成立，因此 username='' OR 1=1  恒成立，因此SQL语句等同于：

## 3.4sql注入解决办法

第一步：编译sql

第二步：执行sql

优点：能预编译sql语句

|  |
| --- |
| **String username = "username=' OR 1=1 -- ";**  **String password = "12345";**  **// String sql = "SELECT id,username FROM user\_table WHERE " +**  **// "username='" + username + "'AND " + "password='"**  **// + password + "'";**  **String sql = "SELECT id,username FROM user\_table WHERE username=? AND password=?";**  **Class.*forName*("com.mysql.jdbc.Driver");**  **Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/test", "root", "root");**  **PreparedStatement stat = con.prepareStatement(sql);**  **stat.setString(1, username);**  **stat.setString(2, password);**  **System.*out*.println(stat.toString());**  **ResultSet rs = stat.executeQuery();**  **while (rs.next()) {**  **String id = rs.getString(1);**  **String name = rs.getString(2);**  **System.*out*.println("id:" + id + "---name:" + name);**  **}** |

## 3.4 mybatis中#与$区别

动态 sql 是 mybatis 的主要特性之一，在 mapper 中定义的参数传到 xml 中之后，在查询之前 mybatis 会对其进行动态解析。mybatis 为我们提供了两种支持动态 sql 的语法：#{} 以及 ${}。

在下面的语句中，如果 username 的值为 zhangsan，则两种方式无任何区别：

select \* from user where name = #{name};

select \* from user where name = ${name};

其解析之后的结果均为

select \* from user where name = 'zhangsan';

　但是 #{} 和 ${} 在预编译中的处理是不一样的。#{} 在预处理时，会把参数部分用一个占位符 ? 代替，变成如下的 sql 语句：

select \* from user where name = ?;

而 ${} 则只是简单的字符串替换，在动态解析阶段，该 sql 语句会被解析成

select \* from user where name = 'zhangsan';

以上，#{} 的参数替换是发生在 DBMS 中，而 ${} 则发生在动态解析过程中。

那么，在使用过程中我们应该使用哪种方式呢？

答案是，优先使用 #{}。因为 ${} 会导致 sql 注入的问题。看下面的例子：

　select \* from ${tableName} where name = #{name}

在这个例子中，如果表名为

 user; delete user; --

　　则动态解析之后 sql 如下：

select \* from user; delete user; -- where name = ?;

　　--之后的语句被注释掉，而原本查询用户的语句变成了查询所有用户信息+删除用户表的语句，会对数据库造成重大损伤，极大可能导致服务器宕机。

但是表名用参数传递进来的时候，只能使用 ${} ，具体原因可以自己做个猜测，去验证。这也提醒我们在这种用法中要小心sql注入的问题。

### 3.4.1创建UserTable

|  |
| --- |
| **package** com.itmayiedu.entity;  **public** **class** UserTable {  **private** **int** id;  **private** String userName;  **private** String passWord;  } |

### 3.4.2创建UserTable

|  |
| --- |
| **pa*ckage com.itmayiedu.mapper;***  ***import com.itmayiedu.entity.UserTable;***  ***public interface UserTableMapper {***  ***public UserTable login(UserTable userTable);***  ***}*** |

### 3.4.3userTableMapper.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.itmayiedu.mapper.UserTableMapper"*>  <select id=*"login"* parameterType=*"com.itmayiedu.entity.UserTable"*  resultType=*"com.itmayiedu.entity.UserTable"*>  SELECT id ,username as userName FROM user\_table WHERE  username=${userName} AND password=${passWord}  </select>  </mapper> |

### 3.4.4 测试SQL注入

|  |
| --- |
| **public class TestLoginMybatis3 {**  **public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {**  **String resource = "mybatis.xml";**  **// 读取配置文件**  **Reader reader = Resources.*getResourceAsReader*(resource);**  **// 获取会话工厂**  **SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);**  **SqlSession openSession = sqlSessionFactory.openSession();**  **// 查询**  **String sql = "com.itmayiedu.mapper.UserTableMapper.login";**  **// 调用api查询**  **UserTable userTable = new UserTable();**  **userTable.setUserName("'' OR 1=1 -- ");**  **userTable.setPassWord("12345");**  **List<UserTable> listUserTable = openSession.selectList(sql, userTable);**  **for (UserTable ub : listUserTable) {**  **System.*out*.println(ub.getUserName());**  **}**  **}**  **}** |

### 3.4.5 总结

优先使用 #{}。因为 ${} 会导致 sql 注入的问题

# Mybatis 注解使用

Mybatis提供了增删改查注解、@select @delete @update

## 4.1 建立注解Mapper

|  |
| --- |
| **import** org.apache.ibatis.annotations.Param;  **import** org.apache.ibatis.annotations.Select;  **import** com.itmayiedu.entity.User;  **public** **interface** UserTestMapper {  @Select("select \* from users where id = ${id};")  public User getUser(@Param("id") String id);  } |

## 4.2 加入mybatis.xml

|  |
| --- |
| <mapper class="com.itmayiedu.mapper.UserTestMapper" /> |

## 4.3 运行测试

|  |
| --- |
| **public class TestMybatis3 {**  **public static void main(String[] args) throws IOException {**  **String resource = "mybatis.xml";**  **// 读取配置文件**  **Reader reader = Resources.*getResourceAsReader*(resource);**  **// 获取会话工厂**  **SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);**  **SqlSession openSession = sqlSessionFactory.openSession();**  **// 调用api查询**  **UserTestMapper userTestMapper=openSession.getMapper(UserTestMapper.class);**  **System.*out*.println(userTestMapper.getUser("2"));**  **}**  **}** |

# 五、Generator使用

Generator 逆向生成 使用