

# 刘沛霞 18600949004

# Day01-JDBC

# 数据库回顾

Mysql数据库的安装：

1. 下载：[www.mysql.com](http://www.mysql.com)
2. 下载python3.7

https://www.python.org/downloads/windows/

1. 下载 visual studio

https://visualstudio.microsoft.com/zh-hans/vs/compare/

1. Windows上安装：

Cmd登录mysql：

mysql –h localhost –u root –p

123456

MySQL 5.7 Command Line Client登录：

下载图像化客户端登录：

SQLyog10.2

## 在jtdb中创建account表，有id,name,money字段

C:\Users\tarena>mysql -uroot –p

mysql> create database jtdb default character set utf8;

mysql> show databases;

mysql> use jtdb;

mysql> create table account (id int

primary key auto\_increment,

name varchar(50),

money double);

mysql> show tables;

## 往account表中插入2条记录

mysql> insert into account values(null,'xionger',1000);

mysql> insert into account values(null,'xiongda',1000);

mysql> select \* from account;

## 在jtdb中创建user表，有id,username,password字段

mysql> create table user(id int primary key

-> auto\_increment,

-> username varchar(100),

-> password varchar(50));

mysql> show tables;

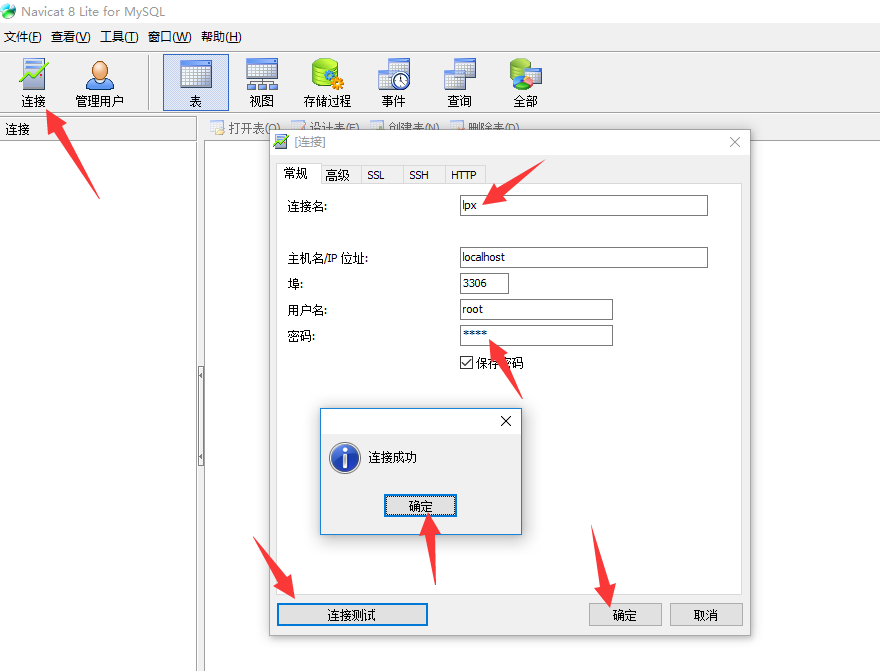
## 在user表中插入2条记录

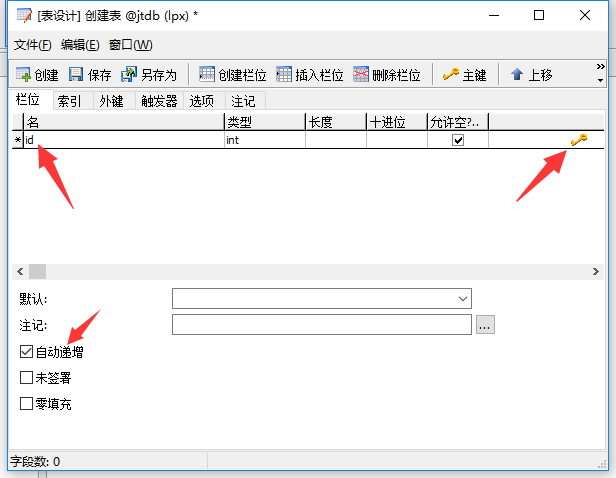
mysql> insert into user values(null,'xiyangyang','123');

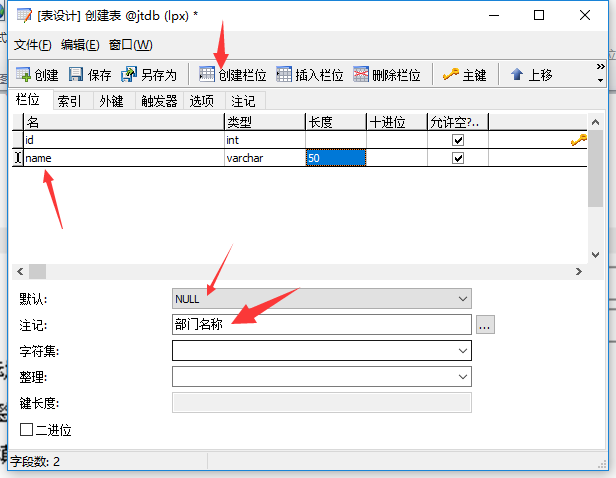
mysql> insert into user values(null,'jack','123');

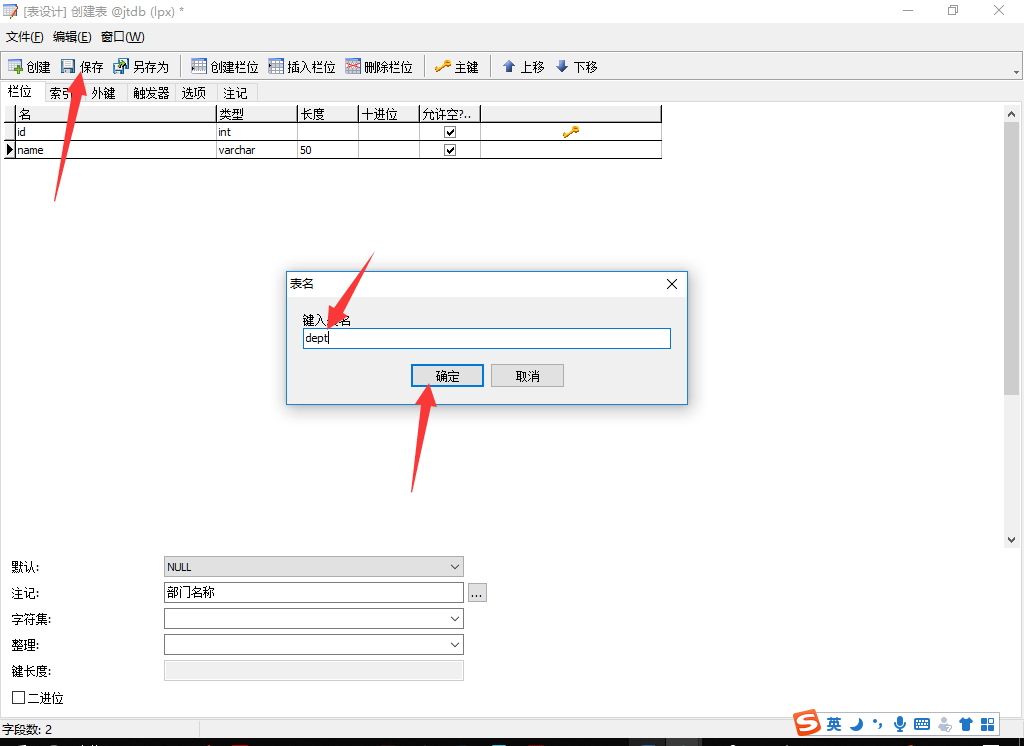
mysql> select \* from user;

## 创建dept表，有id,name字段

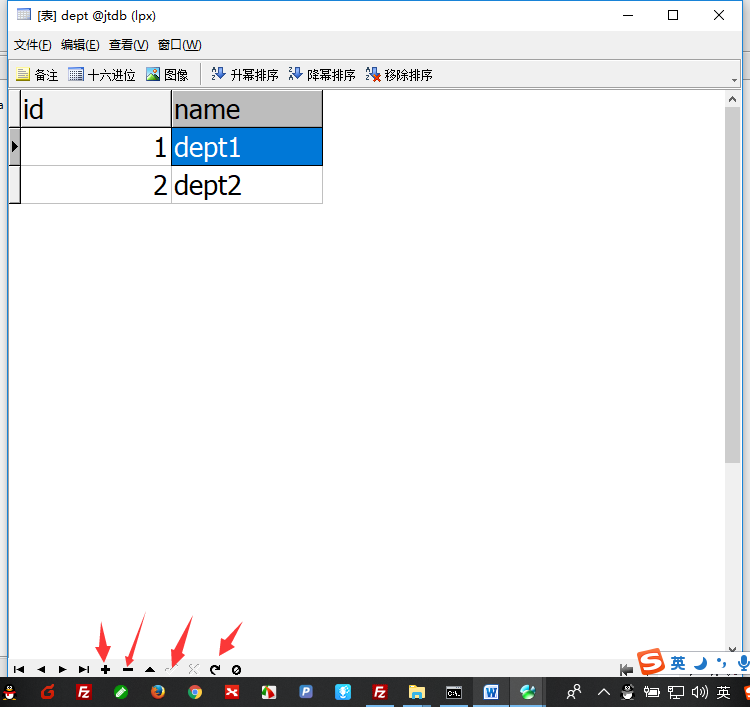




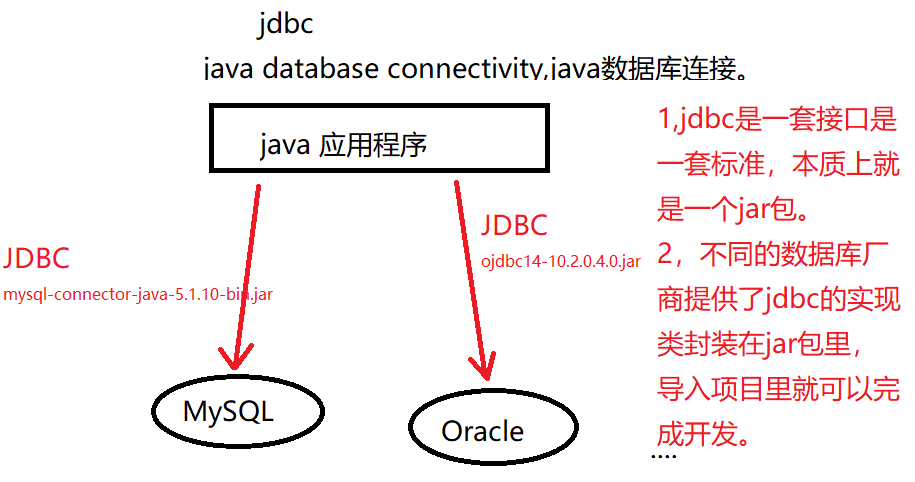




## 在dept表中插入2条记录



# JDBC原理



# JDBC概述

## 什么是jdbc

Jdbc是简称，java database connectivity,java数据库连接，专门用来通过一段java代码连接数据库的一门技术。

后面会学习操作数据库的框架MyBatis，Hibernate，底层都是封装了jdbc。

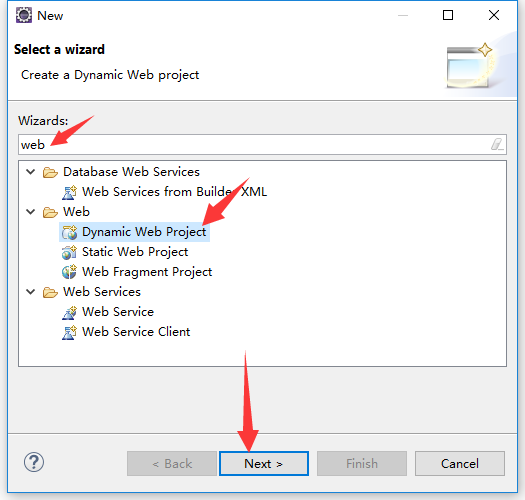
## 如何使用jdbc

### 需求：

通过jdbc技术，查询user表中所有的记录。

### 开发步骤

#### 创建webday01工程



#### 创建HelloJDBC类

#### 创建单元测试方法，hello



#### 代码

**package** cn.tedu.hello;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** org.junit.Test;

**import** com.mysql.jdbc.Driver;

**public** **class** HelloJDBC {

//单元测试方法：@Test + void

@Test

**public** **void** hello() **throws** SQLException{

// 1，注册驱动com.mysql.jdbc.Driver

DriverManager.*registerDriver*(**new** Driver());

// 2，获取数据库连接java.sql.Connection

Connection conn =

DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/jtdb",

"root","root");

// 3，获取传输器java.sql.Statement

Statement st = conn.createStatement();

// 4，执行SQL。java.sql.ResultSet

String sql ="select \* from user";

ResultSet rs = st.executeQuery(sql);

// 5，遍历结果集

**while**(rs.next()){

//根据列的索引获取第一列的数据

String id = rs.getString(1);

//根据列的索引获取第二列的数据

String username = rs.getString(2);

//根据列的索引获取第三列的数据

String password = rs.getString(3);

System.***out***.println(id+username+password);

}

// 6，释放资源

rs.close();//释放结果集资源

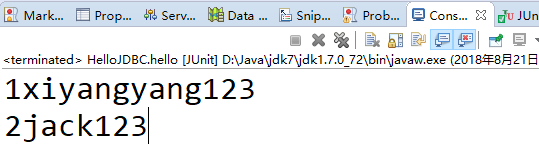
st.close();//释放传输器资源

conn.close();//释放连接资源

}

}

#### 测试

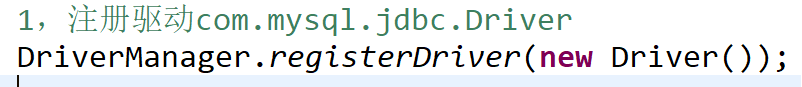


# JDBC API详解

## 注册驱动

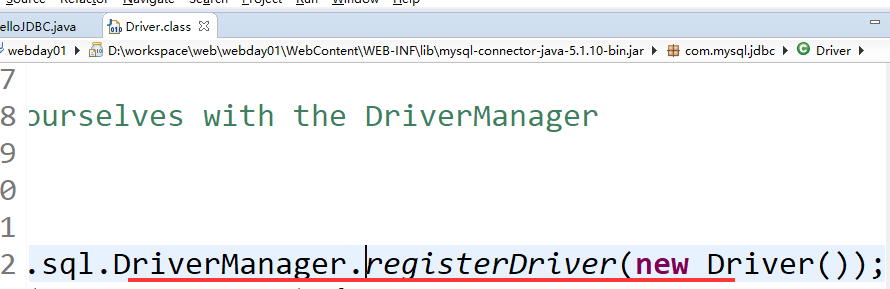
### 作用

将java代码和数据库程序产生关系



### 缺点

1. 注册两次驱动



1. 产生了紧耦合的关系，

[com](eclipse-javadoc:%E2%98%82=webday01/D:%5C/workspace%5C/web%5C/webday01%5C/WebContent%5C/WEB-INF%5C/lib%5C/mysql-connector-java-5.1.10-bin.jar%3Ccom).[mysql](eclipse-javadoc:%E2%98%82=webday01/D:%5C/workspace%5C/web%5C/webday01%5C/WebContent%5C/WEB-INF%5C/lib%5C/mysql-connector-java-5.1.10-bin.jar%3Ccom.mysql).[jdbc](eclipse-javadoc:%E2%98%82=webday01/D:%5C/workspace%5C/web%5C/webday01%5C/WebContent%5C/WEB-INF%5C/lib%5C/mysql-connector-java-5.1.10-bin.jar%3Ccom.mysql.jdbc).[Driver](eclipse-javadoc:%E2%98%82=webday01/D:%5C/workspace%5C/web%5C/webday01%5C/WebContent%5C/WEB-INF%5C/lib%5C/mysql-connector-java-5.1.10-bin.jar%3Ccom.mysql.jdbc(Driver.class%E2%98%83Driver).Driver() throws [SQLException](eclipse-javadoc:%E2%98%82=webday01/D:%5C/workspace%5C/web%5C/webday01%5C/WebContent%5C/WEB-INF%5C/lib%5C/mysql-connector-java-5.1.10-bin.jar%3Ccom.mysql.jdbc(Driver.class%E2%98%83Driver~Driver%E2%98%82java.sql.SQLException)

### 解决方案

Class.forName(“com.mysql.jdbc.Driver”);

## 数据库的URL

### 作用

1. 指定要访问的数据库服务器和数据库的名字

jdbc:mysql://localhost:3306/jtdb

---------- ============== ~~~

协议名 ip地址：端口号 数据库名字

### 简写形式

jdbc:mysql:///jtdb

前提：必须访问本地的数据库服务器localhost+必须使用默认的端口3306（不提倡）

## 数据库连接对象Connection

### 作用

通过DriverManager对象获取到的，需要传入三个参数，数据库的url，数据库的用户名，用户名的密码。返回代表数据库连接的对象Connection。利用Connection可以获取到传输器对象。

### 常用方法

CreateStatement() --- 返回传输器对象

PrepareStatement（） -- 返回带有预编译效果的传输器对象

## 传输器对象Statement

### 作用

用来执行SQL，如果执行的是查询的SQL，返回一个结果集对象。

### 常用方法

executeQuery() --- 用来执行查询的SQL

executeUpdate() -- 用来执行增删改的SQL

## 结果集对象ResultSet

### 作用

用来保存SQL执行完的数据。

### 常用方法

Next() --- 类似于指针的效果，会向下移动

GetString(int index) --- 根据索引获取指定列的值

GetString(String columnname) --- 根据列明获取指定列的值

GetInt(int index) --- 根据索引获取指定列的值

GetInt(String columnname) --- 根据列明获取指定列的值

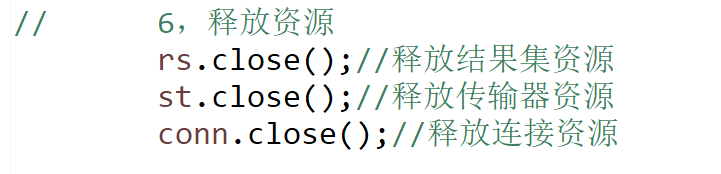
…….

## 关闭资源

### 需求

在jdbc的开发中，释放资源的过程是必须要保证完成的，jdbc资源非常稀缺。在释放资源的过程中，会发生异常，这个时候为了保证资源一定会被释放，需要把释放资源的代码放在finally语句块中，来保证finally块中的代码一定会被执行到。

### 现状



### 改造

}**finally**{//保证资源一定会被释放

// 6，释放资源

//释放结果集资源

**if**(rs!=**null**){//防止空指针异常

**try** {

rs.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

rs=**null**;//手动释放资源

}

}

//释放传输器资源

**if**(st!=**null**){//防止控指针异常

**try** {

st.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

st=**null**;//手动释放资源

}

}

//释放连接资源

**if**(conn!=**null**){//防止控指针异常

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

conn=**null**;//手动释放资源

}

}

}

# JDBC的增删改查

## 新增

### 需求

利用jdbc的技术，向account表中插入一条记录。账户名是王海涛，余额1000。

### 代码

**package** cn.tedu.crud;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** org.junit.Test;

/\*\*

\* 这个类用来完成jdbc的增删改查业务

\*/

**public** **class** JdbcCrud {

//单元测试方法 @Test + void + public

@Test

**public** **void** add(){

Connection conn = **null**;//声明连接对象

Statement st = **null**;//声明传输器对象

**try** {

// 1，注册驱动

Class.*forName*(

"com.mysql.jdbc.Driver");

// 2，获取数据库连接java.sql.Connection

String url = "jdbc:mysql:///jtdb";//数据库的连接地址

String user = "root";

String password = "root";

conn =

DriverManager.*getConnection*(

url, user, password);

// 3，获取传输器java.sql.Statement

st = conn.createStatement();

// 4，执行SQL

String sql =

"insert into account values(null,'taoge',1000)";

**int** rows = st.executeUpdate(sql);

// 5，遍历结果集

System.***out***.println(rows);

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

//6，释放资源st,conn

**if**(st!=**null**){//非空判断

**try** {

st.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

st=**null**;//手动置空

}

}

**if**(conn!=**null**){//非空判断

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

conn=**null**;//手动置空

}

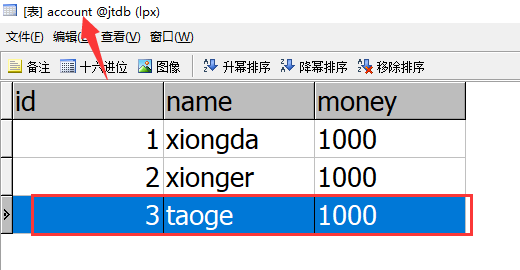
}

}

}

}

### 测试



## 修改

### 需求

利用jdbc的技术，修改account表里name是taoge的记录，将money值改为1001。

### 代码

/\*

\* 需求：利用jdbc的技术， 修改account表里name是taoge的记录， 将money值改为1001

\*/

// 单元测试方法update

@Test

**public** **void** update() {

Connection conn = **null**;//声明连接对象

Statement st = **null**;//声明传输器对象

**try** {

// 1，注册驱动

Class.*forName*("com.mysql.jdbc.Driver");

// 2，获取数据库连接

String url = "jdbc:mysql:///jtdb";//声明数据库连接

String user = "root";//声明用户名

String password = "root";//声明密码

conn =

DriverManager.*getConnection*(

url, user, password);

// 3，获取传输器

st = conn.createStatement();

// 4，执行SQL

String sql =

"update account set money=1001 where name='taoge'";

**int** rows = st.executeUpdate(sql);

// 5，遍历结果集

System.***out***.println(rows);

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

// 6，释放资源st,conn

**if**(st!=**null**){//非空判断

**try** {

st.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

st=**null**;//手动置空

}

}

**if**(conn!=**null**){//非空判断

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

conn=**null**;//手动置空

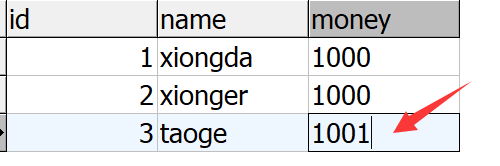
}

}

}

}

### 测试



## 删除

### 需求

利用jdbc技术，删除account表中id为3的记录

### 代码

/\*

\* 利用jdbc技术，删除account表中id为3的记录

\*/

@Test

**public** **void** del() {

Connection conn = **null**;//声明连接对象

Statement st = **null**;//声明传输器对象

**try** {

// 1，注册驱动

Class.*forName*("com.mysql.jdbc.Driver");

// 2，获取数据库连接

String url="jdbc:mysql:///jtdb";

String user="root";

String password="root";

conn = DriverManager.*getConnection*(

url, user, password);

// 3，获取传输器

st = conn.createStatement();

// 4，执行SQL

String sql =

"delete from account where id=3";

**int** rows = st.executeUpdate(sql);

// 5，遍历结果集

System.***out***.println(rows);

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

// 6，释放资源st,conn

**if**(st!=**null**){//非空判断

**try** {

st.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

st=**null**;//手动置空

}

}

**if**(conn!=**null**){//非空判断

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

conn=**null**;//手动置空

}

}

}

}

### 测试

数据库中id为3的记录成功被删除！！

# JDBCUtils工具类

## 私有化构造函数

## 提供公共的，静态的，getConnection 方法，用来给外界提供数据库连接

## 提供公共的，静态的，close方法，用来释放资源

## 测试类

# 作业

1、使用 Statement 对象完成对 jtdb 数据库中 user 表的增删改查操作

(1) 查询 jtdb 数据库中 user 表中的的数据，并打印在控制台上。

(2) 插入一个 username 为 tony，password 为 123 的用户。

(3) 将 tony 用户的密码修改为 '123456'。

(4) 将 tony 用户从 user 表中删除。