VS2010 编程环境下用 MFC 连接 Mysql 数据库

小编近来由于工作需要，需要折腾一下MySql据库, 由于以前用的数据库不是 MySql, 也不是在 VS2010 , 更不是在 MFC 中连接数据库。 所以还颇费了一番周折才实现了把数据插入到数据库表格， 以及把数据从服务器中取并简单地显示出来这两项功能。

写这篇文章的目的， 就是想把我的经验分享给大家， 希望能够帮助那些对于 VS2010 + MFC + MYSQL 这个组合不太熟悉的小伙伴们， 使他们能够快速的连接到数据库并对其进行简单的操作。

废话不多说，言归正传。

/////////////////////////////////////////////////////////////////////////////////////////////////

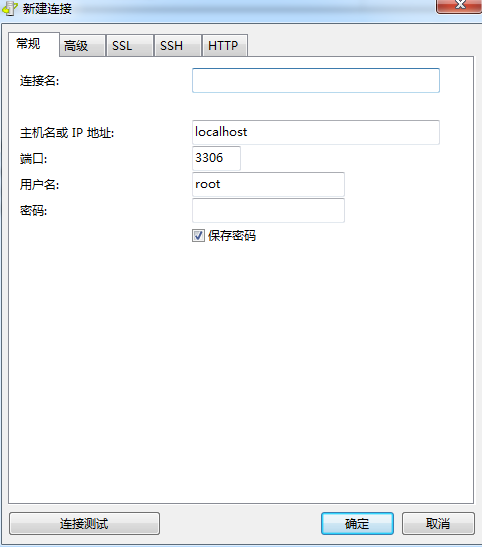
VS2010 和 Mysql 的安装我就不介绍了， 网上很多详细介绍。 ------ 其实我也不会， 是同事帮我安装的。 ------ 大公司就是好， 很多东西都不需要自己动手，有专业的队友帮你搞定。

////////////////////////////////////////////////////////////////////////////////////////////////

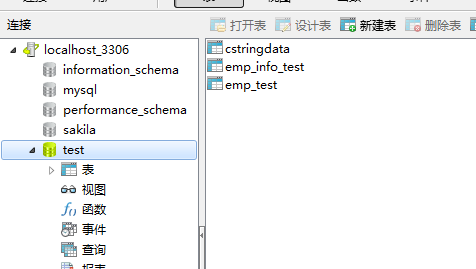
1. 打开Navicat for MySql, 新建一个数据库 test, 在数据库中新建一张表格 emp\_test， 这个表格包括三列： id, name, salary, 数据类型都是 varchar, 长度都是200 。

如图：

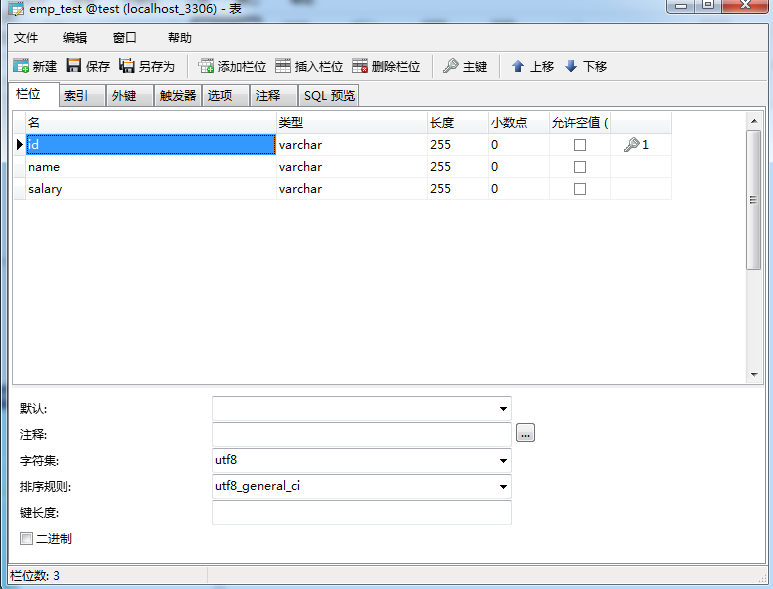
1. 数据库的连接设置：



1. 新建的数据库及表：



1. 新建的表及设置：



这个表的每一行用来存放一条员工信息， 分别是：id, name, salary.

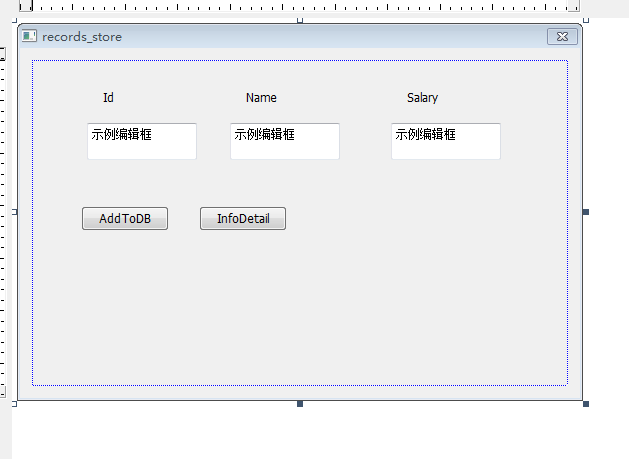
二、新建一个 MFC 的基于对话框的项目。

项目名我起的是： records\_store, 界面效果如下图：

三个文本框的作用是接受用户输入需要上传至服务器的信息。

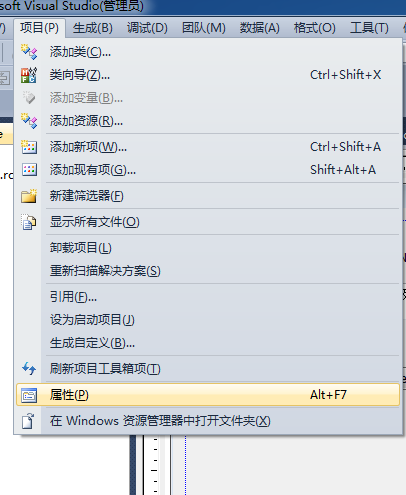
按键 AddToDB 的作用是： 当我的程序写好之后， 一点击它， 就会将用户在文本框中输入的 id, name, salary 上传到数据库 test 中的表格 emp\_test 中。

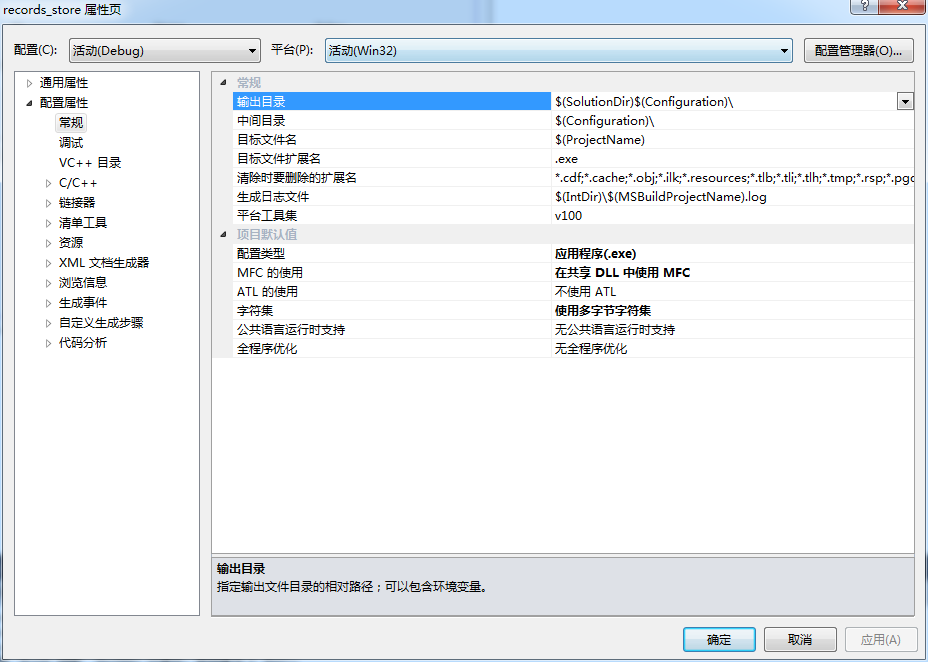
按键 InfoDetail 的作用是： 当我的程序写好之后，用户可以在Id 标签下面的那个文本框中填写一个 id， 然后点击 InfoDetail， 就会在消息框中显示这个 id 的所有信息。



三、 对项目进行配置， 使其可以通过代码连接 MySQL 数据库。

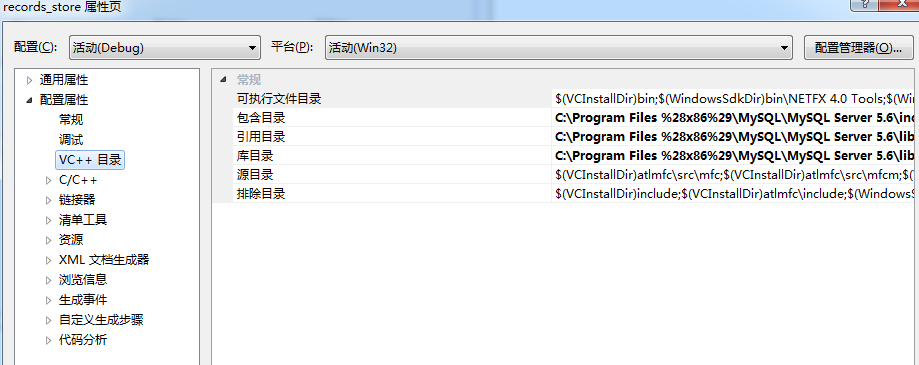
打开属性配置界面。 点击菜单栏中的项目， 在弹出菜单中点击属性，打开属性配置页面。 如图：





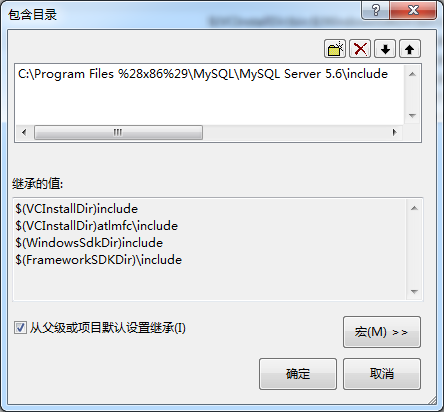
第一步：对包含目录，引用目录和库目录进行设置：

点击 配置属性左边的三角箭头， 展开其子选项， 点击 vc++ 目录， 如下图

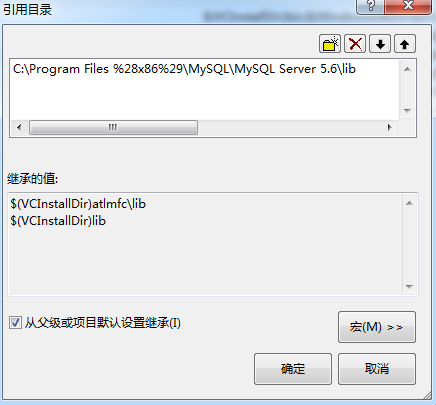


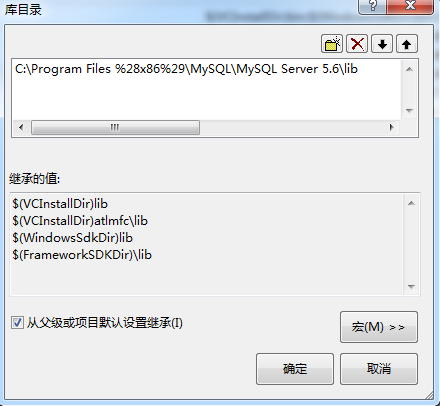
这里有三个目录需要配置， 分别是包含目录，引用目录和库目录。

首先， 对包含目录进行配置。 选中包含目录后， 右边会出现下拉箭头， 点击该箭头，再点击 ‘编辑’，把你电脑里 MySQL 安装目录中的 include 文件的路径填写在编辑框里面（记得点确定哦）。我的 MySQL 是安装在 C盘的， 路径如图：



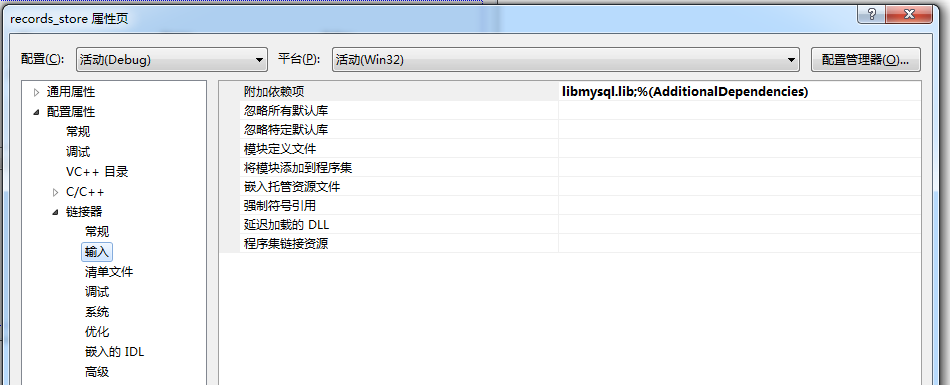
然后， 对引用目录和库目录进行配置。 分别在引用目录和包含目录的编辑框中填写你电脑里MySQL 安装目录中的 lib 文件的路径， 如图所示：



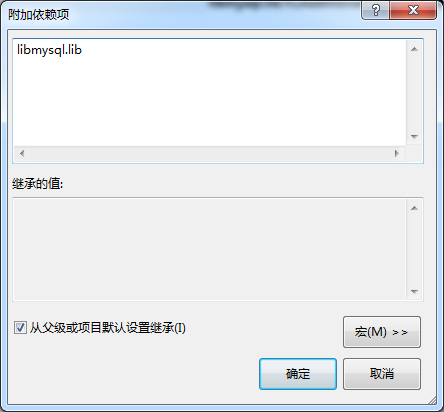


第二步：对附加依赖项进行设置。

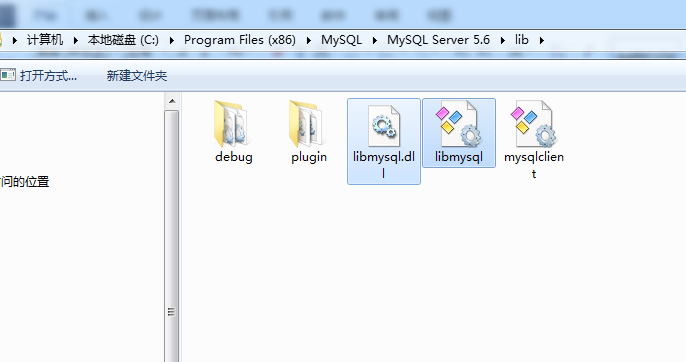
点击 配置属性左边的三角箭头， 展开其子选项后，点击 链接器 左边的三角箭头展开连接器的子选项， 然后点击 输入。然后对 附加依赖项进行进行设置。如图：



注意只要将 libmysql.lib 填到附加依赖项的编辑框中即可， 如图：



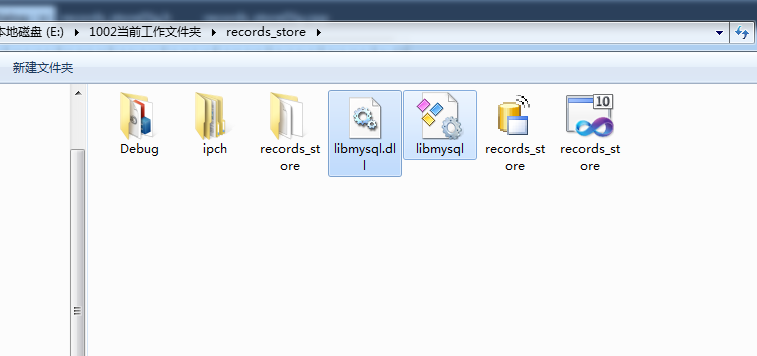
第三步： 将 MySql 安装目录中 libmysql.dll 和 libmysql.lib 两个文件拷贝到当前项目 records\_store 的主目录下， 以及主目录下 的 与项目名同名的文件夹（records\_store文件夹）下。如图：



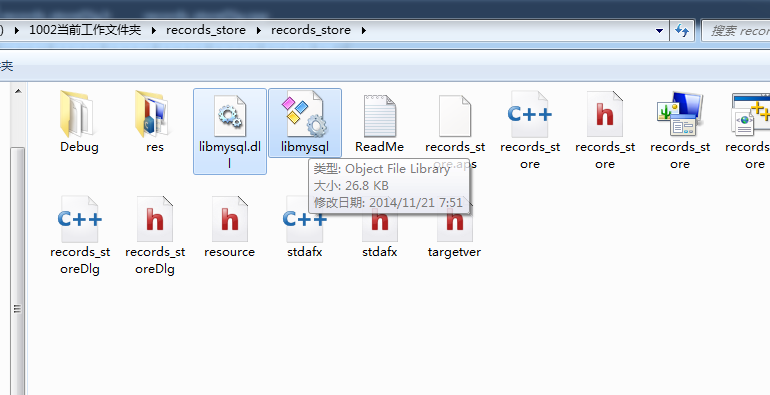
上图的路径是 MySQL 的安装路径。

将libmysql.dll 和 libmysql.lib 两个文件从上图中的路径分别拷贝至下面两个图中的路径中。

下图是项目文件夹。



下图中的路径是项目文件夹下，与项目名同名的文件夹。

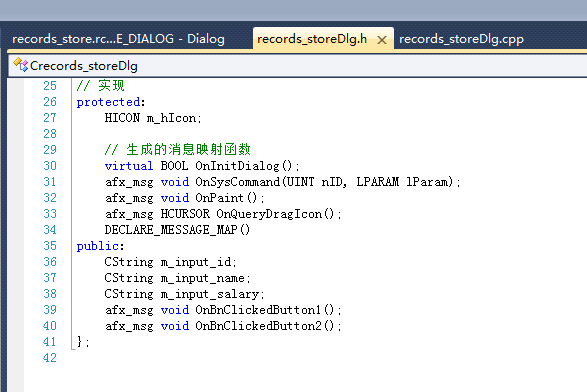


//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**这步做完后， 你就可以在当前项目中使用代码连接数据库了。**

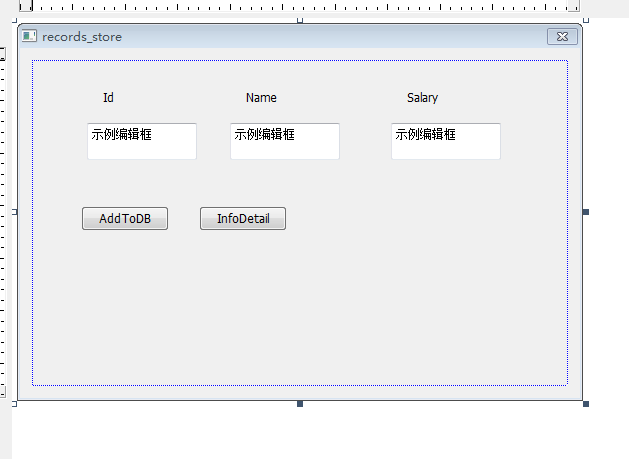
继续写代码。

四、分别给上面创建的对话框界面中的三个编辑框添加变量， 变量名分别为 m\_input\_id, m\_input\_name, m\_input\_salary, 变量类型都写成 CString. 最后生成的变量声明代码如下图：



/////////////////////////////////////////////////////////////////////////////////////////////////

这里再贴一张刚才做的界面图。



五、数据上传按键 AddToDB 的单击消息处理函数的实现：

void Crecords\_storeDlg::OnBnClickedButton1()

{

UpdateData(TRUE);

CString local\_id = m\_input\_id.GetBuffer();

CString local\_name = m\_input\_name.GetBuffer();

CString local\_salary = m\_input\_salary.GetBuffer();

// 创建 MYSQL 对象。

MYSQL local\_mysql;

// 初始化 MYSQL 对象。

mysql\_init(&local\_mysql);

// 连接数据库

if(!mysql\_real\_connect(&local\_mysql,"localhost","root","123456","test",3306,NULL,0))

{

AfxMessageBox(\_T("connect to database failed!"));

}

else

{

AfxMessageBox(\_T("connect to database success!"));

// 设置字符集， 使程序支持中文。

mysql\_query(&local\_mysql, "set names 'gb2312'");

}

// 创建 SQL 语句字符串

CString sql\_insert;

sql\_insert.Format("insert into emp\_test(id, name, salary) values (\'%s\', \'%s\', \'%s\')", local\_id, local\_name, local\_salary);

// 执行 sql 语句。

// mysql\_query() 的返回值份很多情形， 进行判断使要注意。

if(mysql\_query(&local\_mysql,(char\*)(LPCTSTR)sql\_insert)==0)

{

AfxMessageBox("insert succeed!");

}else{

AfxMessageBox("insert failed!");

}

// 关闭 local\_mysql 所关联的数据库连接， 一般情况下不用写。

mysql\_close(&local\_mysql);

UpdateData(FALSE);

}

六、数据查找按键 InfoDetail 的单击消息处理函数的实现：

void Crecords\_storeDlg::OnBnClickedButton2()

{

UpdateData(TRUE);

// 将用户在编辑框的输入放入变量， 这里我只实现了通过 id 查找。

CString local\_id = m\_input\_id.GetBuffer();

//CString local\_name = m\_input\_name.GetBuffer();

//CString local\_salary = m\_input\_salary.GetBuffer();

// 创建 MYSQL 对象。

MYSQL local\_mysql;

// 初始化 MYSQL 对象。

mysql\_init(&local\_mysql);

// 连接数据库。

if(!mysql\_real\_connect(&local\_mysql,"localhost","root","123456","test",3306,NULL,0))

{

AfxMessageBox(\_T("connect to database failed!"));

}

else

{

AfxMessageBox(\_T("connect to database success!"));

// 设置字符集， 使程序支持中文。

mysql\_query(&local\_mysql, "set names 'gb2312'");

}

//CString select\_sql\_all;

CString select\_sql\_by\_id;

//CString select\_sql\_by\_name;

// 存放查询结果的变量

MYSQL\_RES\* result;

MYSQL\_ROW row;

//select\_sql\_all.Format("select \* from emp\_test");

select\_sql\_by\_id.Format("select id, name, salary from emp\_test where id = \'%s\'", local\_id.GetBuffer());

// select\_sql\_by\_name.Format("select id, name, salary from emp\_test where name = \'%s\'", local\_name.GetBuffer());

int res = mysql\_query(&local\_mysql, (char\*)(LPCSTR)select\_sql\_by\_id);

// mysql\_query 函数返回 0 表示 查询成功。

if(res==0){

// 获取查询结果，并保存在 result 里面。

result = mysql\_store\_result(&local\_mysql);

// 获取查询结果的行数， 这里用不到，所以注释了。

// int row\_count = mysql\_num\_rows(result);

// 循环地从查询结果中拿出一行， 并赋值给 row 变量。

while(row = mysql\_fetch\_row(result)){

CString local\_id;

CString local\_name;

CString local\_salary;

CString local\_message;

// 将从数据库中获取的数据 保存在字符串中， row[0], row[1], row[2], 分别表示一条查询结果中的第1， 2， 3 个字段， 我建立的 emp\_test

// 表中只有 3 个字段。

local\_id.Format(row[0]);

local\_name.Format(row[1]);

local\_salary.Format(row[2]);

// 将字符串连接起来，供消息框显示。

local\_message.Format("results: ");

local\_message += local\_id + \_T(",") +local\_name + \_T(",") + local\_salary;

AfxMessageBox(local\_message.GetBuffer());

}

mysql\_free\_result(result);

AfxMessageBox("select succeed!");

}else{

AfxMessageBox("no results!");

}

mysql\_close(&local\_mysql);

UpdateData(FALSE);

}

介绍到这里， 基本就介绍完了，顺便附上 mysql\_real\_connect 函数的具体说明：

/\*

// 连接数据库。

mysql\_real\_connect(&local\_mysql, // MYSQL \*mysql, MYSQL 对象。

"localhost", // const char \*host,该参数必须是主机名或IP地址。如果传入的参数是是 NULL 或字符串 "localhost" ，连接将被视为与本地主机的连接。

"root", // const char \*user, MYSQL 用户登录名。

"123456", // const char \*passwd, 密码。

"test", // const char \*db, 数据库名。

3306, // unsigned int port, 端口号。如果“port”不是0，其值将用作TCP/IP连接的端口号。注意，“host”参数决定了连接的类型。

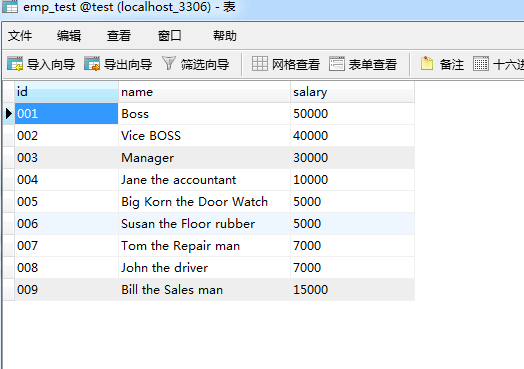
NULL, // const char \*unix\_socket, 如果unix\_socket不是NULL，该字符串描述了应使用的套接字或命名管道。注意，“host”参数决定了连接的类型。

0); // unsigned long client\_flag, client\_flag的值通常为0, 但也可以设置成其他 FLAG 标识的组合。

\*/

这是我试运行的效果：

这是我用我写的对话框程序上传的数据：



下面是查询功能的运行结果：

