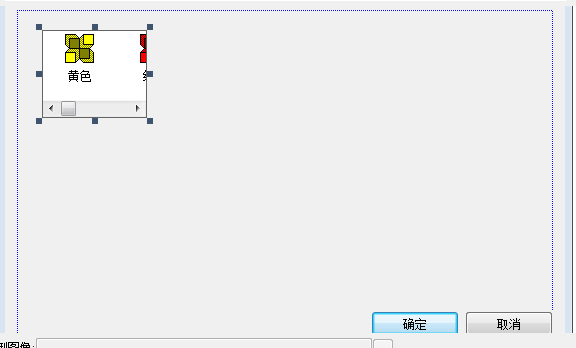
C++连接数据库

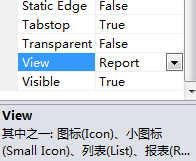
C++连接数据库有很多种方法，ODBC，ADO等等。我这里就采用了ADO的方法。

既然都涉及到数据库了，就直接做个带界面的吧。先打开VS2010，新建一个“MFC应用程序”，在里面记得选择“基于对话框”，因为这个比较简单，我就用这个举例了，其他的自己研究一下吧，然后一直下一步就可以了。

下面正式开始，先把对话框上的静态文本控件删了，没有什么用。再添加一个“List Control”控件，现在应该是这个样子的。

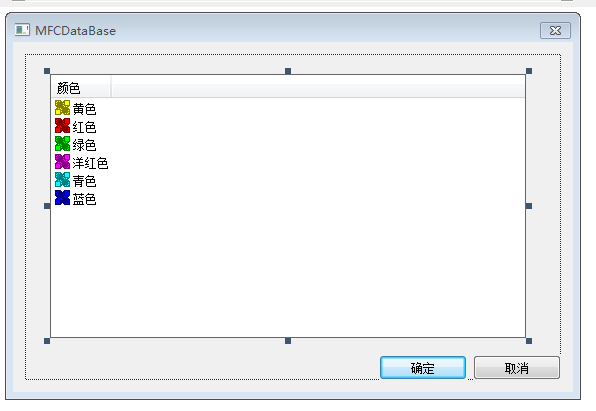


然后右键单击这个List Control，选择属性。在属性里找到View，改成Report，如图：



然后调整一下控件的大小吧。

现在应该是这个样子的。



好了，界面的设计就到这吧，毕竟我不是来讲界面设计的。

然后是数据库，我们需要一个测试用的数据库，我随便设计了一个，用的是SqlServer2005，其他的类似，执行SQL语句：

use master

create database db\_test

use db\_test

create table student(

stu\_id int identity(1,1) primary key,

stu\_num varchar(8) not null default '00000000',

stu\_name varchar(20) not null,

stu\_class varchar(50) not null

)

insert into student values( '20101611', '测试1', '计算机')

insert into student values( '20101612', '测试2', '数学')

insert into student values( '20101613', '测试3', '计算机')

insert into student values( '20101614', '测试4', '中文')

上面的SQL语句的结果是创建了一个名称为db\_test的数据库，并且建立了一个表，名称为student，里面有四列，第一列id为自增。后面添加了四个测试数据。

下面是程序代码部分：

添加一个类，我叫做CDataBaseADO，在DataBaseADO.h文件中的#pragma once后加上一句

#import "C:\Program Files\common files\system\ado\msado15.dll" no\_namespace rename("EOF","adoEOF")

因为ADO是COM组件的一个，这句是引入一个库文件，否则的话，下面的都不能使用。

然后在类里添加几个变量:

private:

// \_ConnectionPtr通常被用来创建一个数据连接或执行一条不返回任何结果的SQL语句，如一个存储过程。

\_ConnectionPtr m\_pConnection;

//记录集

\_RecordsetPtr m\_pRecordset;

数据类型 \_ConnectionPtr实际上就是由类模板\_com\_ptr\_t而得到的一个具体的实例类，其定义可以到msado15.tlh、comdef.h 和comip.h这三个文件中找到。在msado15.tlh中有：

\_COM\_SMARTPTR\_TYPEDEF(\_Collection, \_\_uuidof(\_Collection));

经宏扩展后就得到了\_ConnectionPtr类。\_ConnectionPtr类封装了Connection对象的Idispatch接口指针，及一 些必要的操作。我们就是通过这个指针来操纵Connection对象。类似地，后面用到的\_CommandPtr和\_RecordsetPtr类型也是这 样得到的，它们分别表示命令对象指针和记录集对象的指针。

然后在构造函数和析构函数里写上如下代码：

CDataBaseADO::CDataBaseADO(void)

{

::CoInitialize(NULL); //初始化OLE/COM库环境

m\_pConnection=NULL;

}

CDataBaseADO::~CDataBaseADO(void)

{

if(m\_pConnection)

m\_pConnection->Close();

m\_pConnection=NULL;

::CoUninitialize(); //释放程序占用的COM 资源

}

然后我们来写打开数据库连接和关闭。数据库只有打开了才能使用。

// 打开数据库连接

bool CDataBaseADO::Open(\_bstr\_t strConnection)

{

if( FAILED( m\_pConnection.CreateInstance(\_\_uuidof(Connection)) ) ) //初始化Connection指针

return false;

try{

m\_pConnection->Open(strConnection, "", "", 0);

}catch(\_com\_error e)

{

AfxMessageBox(e.Description());

return false;

}

return true;

}

// 关闭数据库连接

void CDataBaseADO::Close(void)

{

if(m\_pConnection)

m\_pConnection->Close();

m\_pConnection=NULL;

}

这些都很简单了，我注释里面也写得很清楚了。在\_ConnectionPtr使用前都需要初始化，然后打开，用完之后关闭。记住Open()和Close()一定要成对出现，一次打开就要有一次关闭，而且不能多了，也不能少了。

下面就是重头戏了，select操作的实现。这个就是执行了一条select语句后，返回一个记录集，然后我们把记录集处理一下，放到一个容器里，而不是返回记录集指针，这样以后我们在用这个类的时候，就可以不用在每个里面都去写那句#import了，类相对更独立一些。

看看代码先：

// 查询

vector<vector<\_variant\_t>> CDataBaseADO::Select(BSTR strSql, vector<\_variant\_t> strName)

{

\_RecordsetPtr pRecordset; //定义数据集对象

vector<vector<\_variant\_t>> vRecord; //这是C++0x新标准，不是VS2010或者不支持新标准的，

//要写成vector< vector<\_variant\_t> >,因为>>会被认为是右移操作符

if( FAILED( pRecordset.CreateInstance(\_\_uuidof(Recordset)) ) ) //初始化Recordset指针

return vector<vector<\_variant\_t>>();

try{

pRecordset->Open(strSql, (IDispatch\*)m\_pConnection, adOpenDynamic, adLockOptimistic, adCmdText);//adOpenDynamic：动态 adLockOptimistic乐观封锁法 adCmdText：文本查询语句

pRecordset->MoveFirst();

while(!pRecordset->adoEOF)//遍历所有记录

{

//取记录字段值

vector<\_variant\_t> vTheValue; //VARIANT数据类型的泛型

for(int i=0; i<strName.size(); ++i)

{

vTheValue.push\_back( pRecordset->GetCollect( strName.at(i) ) );//得到字段的值,并添加到容器的最后

}

vRecord.push\_back(vTheValue);

pRecordset->MoveNext(); //移动到下一条记录

}

pRecordset->Close(); //关闭连接

pRecordset = NULL;

}catch( \_com\_error e)

{

AfxMessageBox(e.Description());

}

return vRecord;

}

到这里有好多东西要说明的。一个一个来说。

\_variant\_t 记录有数据的类型和数据的值，它封闭了VARIANT数据类型,VARIANT是一个结构体类型，具体的定义可以上网搜一下，也可以看MSDN，我就不介绍了。简单说，它可以是任何类型，在它里面的vt属性表示了它的类型。具体怎么使用，在后面会说到，现在还没有涉及到转化问题。

Vector是一个容器，一个能够存放任意类型的动态数组，能够增加和压缩数据。你可以随时增加或减少里面的数据。因为我们从数据库中获取的数据数量是未知的（在你运行完程序之前）所以要用动态数组来存储。它的定义方法为vector<类型> 变量名，而我们的函数的返回类型呢，是容器的容器，相当于二维数组吧，因为记录集就是那样的，有行有列。里面的是每列的内容，外面的容器存储的是行的内容，每行里面都有需要的列数。

还有一点，关于C++0x的标准问题。因为VS2010是支持C++0x的，所以我直接写成了vector<vector<\_variant\_t>>这样的，>>不会被认为成右移运算符，但是不支持新标准的编译器里这样写就是错误的，因为>>会被认为成右移运算符，所以要在中间加空格，成为vector< vector<\_variant\_t> >这样的。关于C++0x也不是我们的重点，大家注意一下自己的编译器，看看支持新标准不。

我们将\_RecordsetPtr类型的数据集对象定义为局部变量，然后初始化，然后打开。Open()函数的原型如下：

HRESULT Recordset15::Open ( const \_variant\_t & Source, const \_variant\_t & ActiveConnection,enum CursorTypeEnum CursorType, enum LockTypeEnum LockType, long Options )

参数说明：

①Source是数据查询字符串

②ActiveConnection是已经建立好的连接（我们需要用Connection对象指针来构造一个\_variant\_t对象)

③CursorType光标类型，它可以是以下值之一,请看这个枚举结构:

enum CursorTypeEnum

{

adOpenUnspecified = -1,///不作特别指定

adOpenForwardOnly = 0,///前滚静态光标。这种光标只能向前浏览记录集，比如用MoveNext向前滚动,这种方式可以提高浏览速度。但诸如BookMark,RecordCount,AbsolutePosition,AbsolutePage都不能使用

adOpenKeyset = 1,///采用这种光标的记录集看不到其它用户的新增、删除操作，但对于更新原有记录的操作对你是可见的。

adOpenDynamic = 2,///动态光标。所有数据库的操作都会立即在各用户记录集上反应出来。

adOpenStatic = 3///静态光标。它为你的记录集产生一个静态备份，但其它用户的新增、删除、更新操作对你的记录集来说是不可见的。

};

④LockType锁定类型，它可以是以下值之一，请看如下枚举结构：

enum LockTypeEnum

{

adLockUnspecified = -1,///未指定

adLockReadOnly = 1,///只读记录集

adLockPessimistic = 2,悲观锁定方式。数据在更新时锁定其它所有动作，这是最安全的锁定机制

adLockOptimistic = 3,乐观锁定方式。只有在你调用Update方法时才锁定记录。在此之前仍然可以做数据的更新、插入、删除等动作

adLockBatchOptimistic = 4，乐观分批更新。编辑时记录不会锁定，更改、插入及删除是在批处理模式下完成。

};

⑤option可以取以下值

adCmdText:表明CommandText是文本命令

adCmdTable:表明CommandText是一个表名

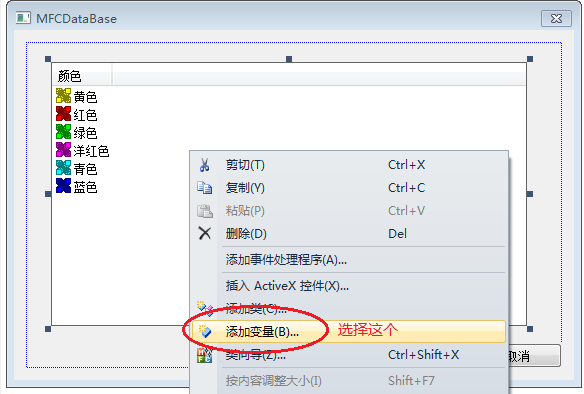
adCmdProc:表明CommandText是一个存储过程

adCmdUnknown:未知

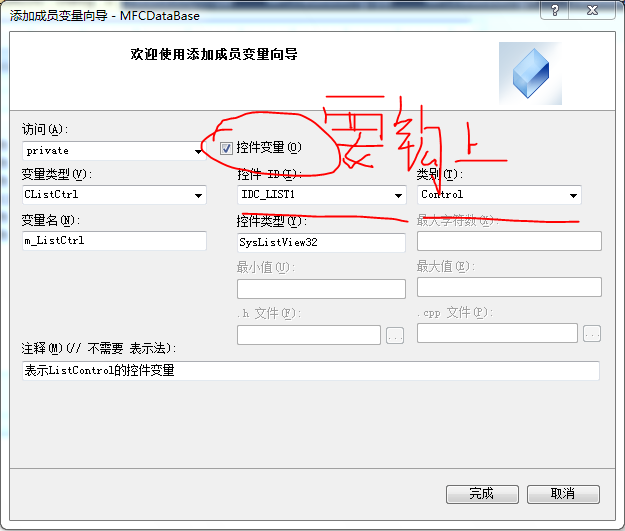
在我们执行了查询操作后，要先将记录集移动到第一个，然后遍历，把所有的结果都放到容器中。遍历结束后，关闭数据集，并且返回。

好吧，到现在为止，数据集已经获取到了，数据也能成功返回了，但是在程序里是显示不出来的，因为我们还没做显示部分。不过这些不是我们的主要内容，附带着讲一下吧。

在资源视图里，打开对话框，然后在ListControl上点右键，添加变量，如图：



在新对话框里，输入名称，我的叫m\_ListCtrl，如图：



当然VC6的要在Class Wizard里设置了。注意的是控件ID是否正确。一般的说，不用设置了，类别选成Control。

然后我们就要将这个ListControl设置成四列，并且设置一些小格式。见代码：

CRect rc;

m\_ListCtrl.GetWindowRect(&rc); //获取控件大小

//设置了四列，大小是一样的

m\_ListCtrl.InsertColumn(0, \_T("序号"), LVCFMT\_CENTER, rc.Size().cx/4, 0);

m\_ListCtrl.InsertColumn(1, \_T("学号"), LVCFMT\_CENTER, rc.Size().cx/4, 1);

m\_ListCtrl.InsertColumn(2, \_T("姓名"), LVCFMT\_CENTER, rc.Size().cx/4, 2);

m\_ListCtrl.InsertColumn(3, \_T("班级"), LVCFMT\_CENTER, rc.Size().cx/4, 3);

//LVS\_EX\_GRIDLINES是希望显示网格；LVS\_EX\_FULLROWSELECT是希望被选中时整行反色显示；LVS\_EX\_HEADERDRAGDROP是让其支持点击表头排序；LVS\_EX\_TWOCLICKACTIVATE是希望有鼠标在未被选中的行上移动的时候有一些效果

m\_ListCtrl.SetExtendedStyle(m\_ListCtrl.GetExtendedStyle() | LVS\_EX\_GRIDLINES | LVS\_EX\_FULLROWSELECT | LVS\_EX\_HEADERDRAGDROP /\*| LVS\_EX\_TWOCLICKACTIVATE\*/);

把这些代码放在BOOL CMFCDataBaseDlg::OnInitDialog()函数的return前，运行就有效果了。

然后，双击一下对话框里的确定按钮，我就不再添加新按钮了。进入到函数里，把里面的OnOK()注释了，添加如下代码：

void CMFCDataBaseDlg::OnBnClickedOk()

{

// TODO: 在此添加控件通知处理程序代码

//CDialogEx::OnOK();

//这里要用到连接字符串，在下面有教怎么找出来

if(!m\_DataBase.Open("Provider=SQLOLEDB.1;Password=aaa;Persist Security Info=True;User ID=sa;Initial Catalog=db\_test;Data Source=YJN-PC\\SQLEXPRESS2005"))

return;

vector<\_variant\_t> vName; //设置要返回的列名

vName.push\_back("stu\_id");

vName.push\_back("stu\_num");

vName.push\_back("stu\_name");

vName.push\_back("stu\_class");

//查询结果

vector<vector<\_variant\_t>> vResult(m\_DataBase.Select(::SysAllocString(L"select \* from student"), vName));

m\_ListCtrl.DeleteAllItems(); //删除所有的项目

//通过循环添加所有的内容

for(int i=0; i<vResult.size(); ++i)

{

m\_ListCtrl.InsertItem(i, VariantToCString( vResult.at(i).at(0) ) ); //插入一行，每行的第一列是序号

m\_ListCtrl.SetItemText(i, 1, VariantToCString(vResult.at(i).at(1)) ); //设置该行的后面列的内容

m\_ListCtrl.SetItemText(i, 2, VariantToCString(vResult.at(i).at(2)) );

m\_ListCtrl.SetItemText(i, 3, VariantToCString(vResult.at(i).at(3)) );

}

m\_DataBase.Close(); //记得要关闭连接

}

看注释应该就明白了吧，嗯，连接字符串的查找方式，有一个简单的方法，随便新建一个文本文档，就是txt的，然后改扩展名，改成udl的，双击打开。在提供程序里选择合适的程序，我选的是“Microsoft OLE DB Provider for SQL Server”，因为我连接的是SQL2005，然后点下一步，根据提示选择，把允许保存密码钩上，选好数据库，测试一下，如果成功了，就OK了。最后点确定。然后把这个文件改回.txt的，用记事本打开，里面就有连接字符串。看看我的是这样的

[oledb]

; Everything after this line is an OLE DB initstring

Provider=SQLOLEDB.1;Password=aaa;Persist Security Info=True;User ID=sa;Initial Catalog=db\_test;Data Source=YJN-PC\SQLEXPRESS2005

第三行就是连接字符串。

还有一个转换函数：

// 转换字符串

CString CMFCDataBaseDlg::VariantToCString(\_variant\_t var)

{

CString str; //转换以后的字符串

switch(var.vt)

{

case VT\_BSTR: //var is BSTR type

str=var.bstrVal;

break;

case VT\_I2: //var is short int type

str.Format(L"%d",(int)var.iVal);

break;

case VT\_I4: //var is long int type

str.Format(L"%d",var.lVal);

break;

case VT\_R4: //var is float type

str.Format(L"%10.6f",(double)var.fltVal);

break;

case VT\_R8: //var is double type

str.Format(L"%10.6f",var.dblVal);

break;

case VT\_CY: //var is CY type

str=COleCurrency(var).Format();

break;

case VT\_DATE: //var is DATE type

str=COleDateTime(var).Format();

break;

case VT\_BOOL: //var is VARIANT\_BOOL

str= (var.boolVal==0) ?L"FALSE": L"TRUE";

break;

default:

str.Format(L"Unk type %d\n",var.vt);

TRACE(L"Unknown type %d\n",var.vt);

}

return str;

}

这个函数实现了将\_variant\_t类型的转换为CString类型。现在来说说吧，看看函数内容大约也明白了吧，根据\_variant\_t里的vt属性可以知道里面的内容是什么类型的，然后对应的显示就OK了。

再来看看添加，修改，删除的实现。

这个非常简单，都在一个函数里就可以实现了。看看代码吧：

// 执行SQL语句，并返回影响的行数

//IsText：是否是文本命令，是：文本命令，否：存储过程

int CDataBaseADO::ExcuteSQL(\_bstr\_t CommandText, bool IsText)

{

\_variant\_t RecordsAffected; //记录影响的行数

try{

if(IsText)

m\_pConnection->Execute(CommandText, &RecordsAffected, adCmdText);

else

m\_pConnection->Execute(CommandText, &RecordsAffected, adCmdStoredProc);

}catch(\_com\_error e)

{

return -1;

}

return RecordsAffected.intVal;

}

这是运用了\_ConnectionPtr里的Execute方法，具体如下：

\_RecordsetPtr Connection15::Execute ( \_bstr\_t CommandText, VARIANT \* RecordsAffected, long Options )

　　其中CommandText是命令字串，通常是SQL命令。

　　参数RecordsAffected是操作完成后所影响的行数,

　　参数Options表示CommandText中内容的类型，Options可以取如下值之一：

　　adCmdText:表明CommandText是文本命令

　　adCmdTable:表明CommandText是一个表名

　　adCmdProc:表明CommandText是一个存储过程

　　adCmdUnknown:未知

我这个函数只能执行SQL语句和存储过程，不过一般也就够用了。具体想要怎么改数据，自己写SQL语句吧，去拼接字符串，然后执行，当然，如果出错了，就返回-1，执行失败了就是0喽。到这就结束了。以后有机会封装个更好的类，这个类可以拿出来用，功能虽然简单，也够用了。测试一下？应该的。

给对话框加个按钮先，添加单击事件，在里面写上如下代码：

void CMFCDataBaseDlg::OnBnClickedButton1()

{

// TODO: 在此添加控件通知处理程序代码

if(!m\_DataBase.Open("Provider=SQLOLEDB.1;Password=aaa;Persist Security Info=True;User ID=sa;Initial Catalog=db\_test;Data Source=YJN-PC\\SQLEXPRESS2005"))

return;

int r = m\_DataBase.ExcuteSQL("insert into student values( '20101616', '测试6', '中文')");

CString str;

str.Format(L"%d", r);

MessageBox( str );

m\_DataBase.Close(); //记得要关闭连接

}

运行一下，弹出个1吧，那就对了。当然了像Open（）这样的，你可以写到别的地方，不用每次都打开，关闭。这些自己发挥吧，我只是抛砖引玉罢了。