**一、ADO概述**  
　　ADO是Microsoft为最新和最强大的数据访问范例 OLE DB 而设计的，是一个便于使用的应用程序层接口。ADO 使您能够编写应用程序以通过 OLE. DB 提供者访问和操作数据库服务器中的数据。ADO 最主要的优点是易于使用、速度快、内存支出少和磁盘遗迹小。ADO 在关键的应用方案中使用最少的网络流量，并且在前端和数据源之间使用最少的层数，所有这些都是为了提供轻量、高性能的接口。之所以称为 ADO，是用了一个比较熟悉的暗喻，OLE 自动化接口。  
　　OLE DB是一组”组件对象模型”(COM) 接口，是新的数据库低层接口，它封装了ODBC的功能，并以统一的方式访问存储在不同信息源中的数据。OLE DB是Microsoft UDA(Universal Data Access)策略的技术基础。OLE DB 为任何数据源提供了高性能的访问，这些数据源包括关系和非关系数据库、电子邮件和文件系统、文本和图形、自定义业务对象等等。也就是说，OLE DB 并不局限于 ISAM、Jet 甚至关系数据源，它能够处理任何类型的数据，而不考虑它们的格式和存储方法。在实际应用中，这种多样性意味着可以访问驻留在 Excel 电子数据表、文本文件、电子邮件/目录服务甚至邮件服务器，诸如 Microsoft Exchange 中的数据。但是，OLE DB 应用程序编程接口的目的是为各种应用程序提供最佳的功能，它并不符合简单化的要求。您需要的API 应该是一座连接应用程序和OLE DB 的桥梁，这就是 ActiveX Data Objects (ADO)。   
**二、在VC中使用ADO**(开发步骤如下：)   
**1**、引入ADO库文件   
　　使用ADO前必须在工程的stdafx.h头文件里用直接引入符号#import引入ADO库文件,以使编译器能正确编译。代码如下所示：   
用#import引入ADO库文件

#import "c:\program files\common files\system\ado\msado15.dll"no\_namespaces rename("EOF" adoEOF")  
　　这行语句声明在工程中使用ADO，但不使用ADO的名字空间，并且为了避免常数冲突，将常数EOF改名为adoEOF。现在不需添加另外的头文件，就可以使用ADO接口了。   
**2**、初始化OLE/COM库环境  
　　必须注意的是，ADO库是一组COM动态库，这意味应用程序在调用ADO前，必须初始化OLE/COM库环境。在MFC应用程序里，一个比较好的方法是在应用程序主类的InitInstance成员函数里初始化OLE/COM库环境。

BOOL CMyAdoTestApp：：InitInstance()  
{  
if(!AfxOleInit())//这就是初始化COM库  
{  
AfxMessageBox(“OLE初始化出错!”);  
return FALSE;  
}   
……   
}  
**3**、ADO接口简介   
　　ADO库包含三个基本接口:\_ConnectionPtr接口、\_CommandPtr接口和\_RecordsetPtr接口。  
　　\_ConnectionPtr接口返回一个记录集或一个空指针。通常使用它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。使用\_ConnectionPtr接口返回一个记录集不是一个好的使用方法。对于要返回记录的操作通常用\_RecordserPtr来实现。而用\_ConnectionPtr操作时要想得到记录条数得遍历所有记录，而用\_RecordserPtr时不需要。   
　　\_CommandPtr接口返回一个记录集。它提供了一种简单的方法来执行返回记录集的存储过程和SQL语句。在使用\_CommandPtr接口时，你可以利用全局\_ConnectionPtr接口，也可以在\_CommandPtr接口里直接使用连接串。如果你只执行一次或几次数据访问操作，后者是比较好的选择。但如果你要频繁访问数据库，并要返回很多记录集，那么，你应该使用全局\_ConnectionPtr接口创建一个数据连接，然后使用\_CommandPtr接口执行存储过程和SQL语句。   
　　\_RecordsetPtr是一个记录集对象。与以上两种对象相比，它对记录集提供了更多的控制功能，如记录锁定，游标控制等。同\_CommandPtr接口一样，它不一定要使用一个已经创建的数据连接，可以用一个连接串代替连接指针赋给\_RecordsetPtr的connection成员变量，让它自己创建数据连接。如果你要使用多个记录集，最好的方法是同Command对象一样使用已经创建了数据连接的全局\_ConnectionPtr接口  
，然后使用\_RecordsetPtr执行存储过程和SQL语句。　   
**4**、使用\_ConnectionPtr接口  
　　\_ConnectionPtr主要是一个连接接口，取得与数据库的连接。它的连接字符串可以是自己直接写，也可以指向一个ODBC　DSN。

\_ConnectionPtr pConn;  
if (FAILED(pConn.CreateInstance("ADODB.Connection")))  
{  
AfxMessageBox("Create Instance failed!");  
return;  
}  
CString strSRC;  
strSRC="Driver=SQL Server;Server=";  
strSRC+="suppersoft";  
strSRC+=";Database=";  
strSRC+="mydb";  
strSRC+=";UID=SA;PWD=";  
CString strSQL = "Insert into student(no,name,sex,address) values(3,'aaa','male','beijing')";  
\_variant\_t varSRC(strSRC);  
\_variant\_t varSQL(strSQL);  
\_bstr\_t bstrSRC(strSRC);  
if (FAILED(pConn->Open(bstrSRC,"","",-1)))  
{  
AfxMessageBox("Can not open Database!");  
pConn.Release();  
return;  
}  
COleVariant vtOptional((long)DISP\_E\_PARAMNOTFOUND,VT\_ERROR);  
pConn->Execute(\_bstr\_t(strSQL),&vtOptional,-1);  
pConn.Release();  
AfxMessageBox("ok!");  
**5**、使用\_RecordsetPtr接口(以连接SQL Server为例)

\_RecordsetPtr pPtr;  
if (FAILED(pPtr.CreateInstance("ADODB.Recordset")))  
{  
AfxMessageBox("Create Instance failed!");  
return FALSE;  
}  
CString strSRC;  
strSRC="Driver=SQL Server;Server=";  
strSRC+="210.46.141.145";  
strSRC+=";Database=";  
strSRC+="mydb";  
strSRC+=";UID=sa;PWD=";  
strSRC+="sa";  
CString strSQL = "select id,name,gender,address from personal";  
\_variant\_t varSRC(strSRC);  
\_variant\_t varSQL(strSQL);  
if(FAILED(pPtr->Open(varSQL,varSRC,adOpenStatic,adLockOptimistic,adCmdText)))  
{  
AfxMessageBox("Open table failed!");  
pPtr.Release();  
return FALSE;  
}  
while(!pPtr->GetadoEOF())  
{  
\_variant\_t varNo;  
\_variant\_t varName;  
\_variant\_t varSex;  
\_variant\_t varAddress;  
varNo = pPtr->GetCollect ("id");  
varName = pPtr->GetCollect ("name");  
varSex = pPtr->GetCollect ("gender");  
varAddress = pPtr->GetCollect ("address");  
CString strNo =(char \*)\_bstr\_t(varNo);  
CString strName =(char \*)\_bstr\_t(varName);  
CString strSex =(char \*)\_bstr\_t(varSex);  
CString strAddress =(char \*)\_bstr\_t(varAddress);  
strNo.TrimRight();  
strName.TrimRight();  
strSex.TrimRight();  
strAddress.TrimRight();  
int nCount = m\_list.GetItemCount();  
int nItem = m\_list.InsertItem (nCount,\_T(""));  
m\_list.SetItemText (nItem,0,strNo);  
m\_list.SetItemText (nItem,1,strName);  
m\_list.SetItemText (nItem,2,strSex);  
m\_list.SetItemText (nItem,3,strAddress);  
pPtr->MoveNext();  
}  
pPtr->Close();  
pPtr.Release();

**6**、使用\_CommandPtr接口  
　　\_CommandPtr接口返回一个Recordset对象，并且提供了更多的记录集控制功能，以下代码示例了使用\_CommandPtr接口的方法：   
　　代码:使用\_CommandPtr接口获取数据

\_CommandPtr pCommand;  
\_RecordsetPtr pRs;  
pCommand.CreateInstance(\_\_uuidof(Command));  
pCommand->ActiveConnection=pConn;  
pCommand->CommandText="select \* from student";  
pCommand->CommandType=adCmdText;  
pCommand->Parameters->Refresh();  
pRs=pCommand->Execute(NULL,NULL,adCmdUnknown);  
\_variant\_t varValue = pRs->GetCollect("name");  
Cstring strValue=(char\*)\_bstr\_t(varValue);  
**7**、关于数据类型转换由于COM对象是跨平台的，它使用了一种通用的方法来处理各种类型的数据，因此Cstring 类和COM对象是不兼容的，我们需要一组API来转换COM对象和C++类型的数据。\_vatiant\_t和\_bstr\_t就是这样两种对象。它们提供了通用的方法转换COM对象和C++类型的数据。