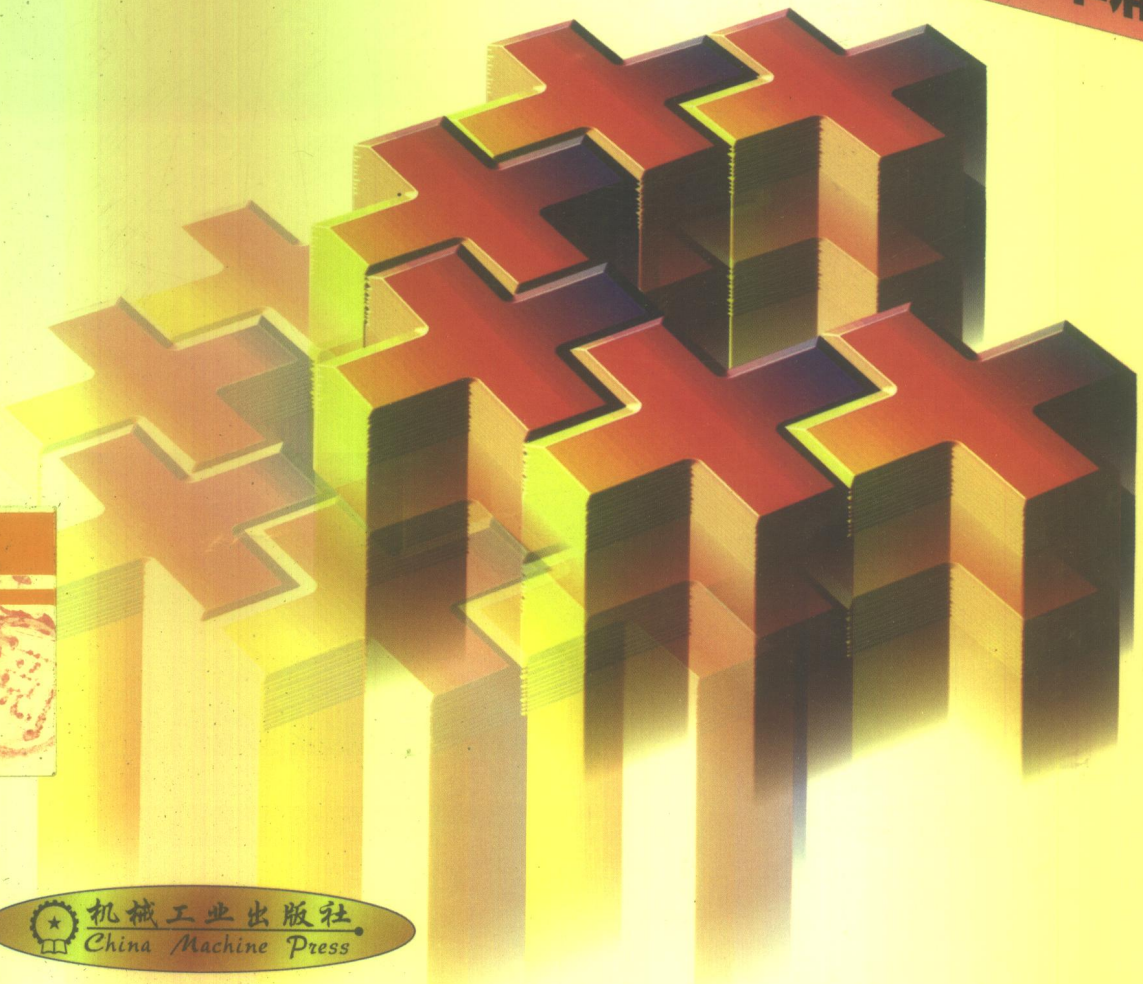
计算机编程技巧

ABC 系列丛书

# C++ Builder 编程技巧

## 网络与数据库篇



清宏计算机工作室 编著

机械工业出版社
China Machine Press

计算机编程技巧 ABC 系列丛书

# C++ Builder 编程技巧

## （网络与数据库篇）

清宏计算机工作室　编著

机 械 工 业 出 版 社

本书以详尽的实例、丰富的内容深入系统地介绍了 C++ Builder 5.0 在网络和数据库方面的编程技巧和方法。针对每一个技巧的主题，分为关键所在（或原理方法）、实现与应用以及专家点评三个部分进行讲解，力求在讲解清楚内容的前提下，缩短篇幅，使读者在有限篇幅下学到更多的技巧。本书按章节编写，各章由既相对独立、又相互关联的技巧主题组成，并且在顺序编排上考虑了前后内容的连续性。

本书适合已懂得 C++并具有 C++ Builder 基本知识的读者学习，可供广大程序员、大专院校师生、计算机爱好者和各种培训班学员参考使用。

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
责任编辑：边　萌　　封面设计：姚　毅
责任印制：郭景龙
北京京丰印刷厂印刷·新华书店北京发行所发行
2001 年 1 月第 1 版第 1 次印刷
787mm×1092mm 1/16·20.75 印张·516 千字
0 001—5 000 册
定价：38.00 元（1CD，含配套书）
ISBN7-900043-44-6/TP·40

# 前　言

随着计算机技术的飞速发展，计算机的开发迅速得到普及推广。当前有一大批计算机开发人员，虽然已经掌握了某种开发工具的基础知识、甚至达到了相当高的水平，但仍有很多盲区，或者急需更多的编程经验和技巧。为此，我们决定组织编写这套“计算机编程技巧 ABC 系列丛书”，以满足广大读者的需求，帮助读者快速成为计算机编程的行家里手。

这套丛书针对当前最流行的几种开发语言，介绍开发过程中的编程经验和技巧，并解决开发过程中的疑点和难点，主要突出“技巧”二字。针对某些开发语言，根据其内容相关性和独立性，我们编写了两个系列图书，即《网络与数据库篇》和《多媒体与系统篇》。这样可以使读者有更大的选择余地，并且在内容编排、读者阅读上都更为合理。

每本书按章节编写，各章由既相对独立，又相互关联的技巧主题组成，并且在顺序编排上考虑了前后内容的连续性。

针对每一个技巧主题，我们将其分为“ABC”三个部分进行讲解。

A——关键所在（或原理方法）　以简单明了的语言指出实现技巧的关键所在或原理方法。

B——实现与应用　以一个简单、完整的实例向读者介绍技巧的实现方法或步骤。

C——专家点评　对技巧加以适当说明，包括补充解释、强调注意事项等。比如简要介绍实现该技巧的其他方法，或者介绍关键技术的扩展应用等。

针对每一个技巧主题，力求在内容讲解清楚的前提下，缩短篇幅，使读者在有限篇幅内学到更多的技巧，整套书充分体现了强调短小精悍、高效率的特点。

读者在阅读本套丛书之前，应该对相应开发语言有了一定了解，但并不要求读者开发水平一定很高，在章节编排和内容讲解过程中，我们尽量照顾了水平还比较低的读者，因此初学者也是可以阅读这套丛书的。

希望这套丛书能使你很快成为软件开发的高手。

恳请广大读者对书中不足之处提出宝贵建议。

清宏计算机工作室

# 编 者 的 话

现今，美国 80%的应用程序使用这样或那样的数据库。事实上，这些程序中的大多数是围绕着如何操作数据库的。进一步讲，客户/服务器数据库如 InterBase、Oracle 和 MS SQL 服务器形成了这些应用程序开发的核心。如果从事软件开发工作，很好地掌握数据库知识是开始编程生涯的最佳途径，现实生活中永远需要优秀的数据库开发人员。学会数据库编程会提供最好的就业机会，特别是客户/服务器数据库编程人员总是供不应求。

本书将分九章，以实例的形式介绍在 C++ Builder 中开发网络和数据库程序的技巧。

第 1 章 “NetMasters 和 WebBroker 控件”中介绍了如何利用 C++ Builder 提供的网络编程控件，实现检测网络的连通性、接受和发送电子邮件、用户查询、创建 Web 浏览器等应用程序。

第 2 章 “网络服务器应用程序”介绍如何编写与 Web 服务器打交道的应用程序，处理 HTML 格式的输入数据。

第 3 章 “Win32 网络编程”介绍了如何使用 WinSock API、WinInet API、NetBIOS 编写网络应用程序，以弥补 C++ Builder 网络编程控件的不足。

第 4 章 “COM、DCOM 和 CORBA”介绍了如何利用 COM、DCOM 和 CORBA 开发网络应用程序。

第 5 章 “数据库存取的连接和设定”介绍了如何通过 C++ Builder 实现数据库、客户机和服务器的连接。

第 6 章 “本地数据库应用系统”首先介绍使用 TTable 对象访问数据库表的一些基本知识，然后介绍了与查询有关的内容以及关系数据库。

第 7 章 “客户/服务器数据库应用系统”介绍了有关使用客户/服务器数据库的技巧，包括数据库的缓冲更新，实现数据库的 Constraint Broker 功能，实现数据库事务处理，以及 MTS 缓冲池等内容。

第 8 章 “分布式多层数据库应用系统”介绍了如何创建分布式多层数据库系统，包括简单的分布式多层数据库应用系统的建立、使用参数动态存取远程数据、客户端应用程序控制数据的存取、使用命令动态存取远程数据，以及多层结构中 Master/Detail 应用系统的实现等内容。

第 9 章 介绍了 WebBroker 和 InternetExpress 这两项技术。

本书不但对 C++ Builder 中网络和数据库编程方面做了全面和详细的介绍，而且通过大量的实例和技巧，结合作者实际开发工作中的经验，对读者可能遇到的一些问题进行了深入透彻的分析，希望能给读者带来一些既切合实际又有用的知识。

**关于光盘**

光盘中包括本书大部分示例的完整代码，它们都是经过严格调试和测试的。

**与作者的联系方式**

zrdl@263.net

由于编者时间关系和水平有限，书中难免出错，希望读者谅解和批评指正。

编 者

# 目　录

# 第 1 章　NetMasters 和 WebBroker 控件

## 1.1　检测网络连通性

### 关键所在

在 Windows 系统中，为测试网络的连通性，常常利用 Windows 系统自带的 Ping.exe 控制台应用程序。其实要检测网络的连通性很简单，只需向某个目的地址发送一个简单的数据包，看能否得到响应即可。C++ Builder 中提供了几个控件用来调试网络，其中一个就是 NMEcho 控件，此控件封装了 echo 协议。它还可以检测网络的速度。

当 TCP/IP 上用户需要建立与服务器之间的连接后，所有发送给服务器的数据都直接发送给客户端。UDP 和 TCP 协议的默认的 echo 端口都是 7。

### 实现与应用

选择 File | New Application，创建一个新项目文件。在 Form1 上加入三个 TPanel 控件，将它们的 Align 属性分别设置为 alTop、alTop 和 alClient，并加入一个 TStatusBar 控件，将属性 SimplePanel 设置为 true。在 Panel1 上加入两个 TLabel，再加入两个 TEdit 控件分别用于设置主机名和超时值。然后再加入一个 TGroupBox 控件，在此控件中放入一个 TEdit 控件(用于设置端口)和一个 TCheckBox(用于设置是否覆盖默认端口)，另外再加入一个 TButton 控件，将它的 Caption 属性设置为回应。在 Panel2 和 Panel3 上分别放置一个 TLabel 控件和一个 TMemo 控件。

双击 Button1 控件，创建此控件的 OnClick 事件处理函数，并加入如下所示的代码：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  //设置标准属性
  NMEcho1->Host = Edit1->Text;
  NMEcho1->TimeOut = StrToInt(Edit2->Text);
  if(CheckBox1->Checked)
    NMEcho1->Port = StrToInt(Edit3->Text);
  else
    NMEcho1->Port = 7;
  NMEcho1->ReportLevel = Status_Basic;
  // 建立连接
```

```
    NMEcho1->Connect();
    Memo2->Clear();
    for(int i=0; i<Memo1->Lines->Count; i++)
      Memo2->Text=Memo2->Text + NMEcho1->Echo(Memo1->Lines->Strings[i]);
    NMEcho1->Disconnect();
  }
  //---------------------------------------------------------------------------
```

然后创建 NMEcho1 控件的一些事件处理函数，并加入如下所示的代码，用于在状态栏中显示此控件当前的状态。

```
  //---------------------------------------------------------------------------
  void __fastcall TForm1::NMEcho1Connect(TObject *Sender)
  {
   StatusBar1->SimpleText="Echo has connected host";
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm1::NMEcho1ConnectionFailed(TObject *Sender)
  {
   ShowMessage("Connection failed");
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm1::NMEcho1Disconnect(TObject *Sender)
  {
   StatusBar1->SimpleText="Echo has disconnected";
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm1::NMEcho1HostResolved(TComponent *Sender)
  {
   StatusBar1->SimpleText="Host resolved";
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm1::NMEcho1InvalidHost(bool &Handled)
  {
   AnsiString TmpStr;
   if(InputQuery("Invalid Host!", "Specify a new host:", TmpStr))    {
       NMEcho1->Host = TmpStr;
       Handled = true;
      }
  }
  //---------------------------------------------------------------------------
```

### 专家点评

程序的核心是 Button1 的 OnClick 事件处理函数。在此函数中，首先从 Host 编辑框控件中读入用户输入的远程主机地址，并将此值赋给 NMEcho 控件的 Host 属性。然后将用户在 TimeOut 编辑框中输入的超时值赋给 NMEcho 控件的 TimeOut 属性。如果不使用默认的端口值，将用户在 Port 编辑框中输入的端口值赋给 NMEcho 控件的 Port 属性，否则应使用默认的端口值 7。在设置好 NMEcho 控件的标准 TCP/IP 属性后，调用此控件的 Connect 方法，建立与 Echo 服务器的连接，将用户在 Input 控件中输入的内容发送给服务器。当服务器接收到客户发送来的数据以后，会将这些数据发送回给客户机，这时，这些数据将显示在 Memo2 控件中。

要注意的是，在调用 NMEcho 控件的 Connect 方法时，应该确保在接收数据之前连接已经建立。如果连接失败，控件将自动抛出异常。但是由于此异常由窗体之外的事件捕捉，所以程序仍然可以正常运行。

当调用 NMEcho 控件的 Connect 方法后，如果用户输入的是域名地址，而不是 IP 地址，并且域名服务器成功地解析了此域名，将触发此控件的 OnHostResolved 事件。如果用户输入的远程主机不正确，将触发此控件的 OnInvalidHost 事件。在此事件处理函数中，本程序显示一个对话框，要求用户重新输入主机的 IP 地址或域名地址。最后，将 Handle 参数设置为 true，这将要求重新试图跟服务器建立连接。

## 1.2　检查主机提供哪些服务

### 关键所在

有些软件，例如 NetXRay，能检查远程主机提供的服务。那么，它们是如何实现的呢？要检查远程主机所提供的服务，关键是以此服务的协议向特定的端口发送一个小包，如果有应答，就表示主机可以提供此服务。如果在特定的时间内没有应答，就表明主机不提供此服务。

C++ Builder 提供了一组 NetMaster 控件。这些控件使用简单，利用它们能在应用程序中实现强大的功能。这组控件在运行期间的特点是当某项操作失败时，抛出异常。因此在程序的实现中，关键是要在程序代码中处理这些异常，而不能让系统弹出对话框说明某项操作失败。例如，如果主机不支持 FTP 服务，当建立连接失败时，系统弹出一对话框说连接失败，因此要在系统之前将此异常处理掉。

### 实现与应用

选择菜单 File | New Application，创建一个新项目文件。在 Form1 上放置三个 TPanel 控件，将它们的 Align 属性分别设置为 alLeft、alLeft 和 alClient。在 Panel1 上放置两个 TLabel

控件和两个TEdit控件，分别用于设置主机地址和超时值，再放入一个TButton控件，将Caption属性设置为“检测”。在Panel2上放置四个TCheckBox控件，将它们的Caption属性分别设置为“检测echo”、“检测DayTime”、“检测Time”和“检测FTP”。在Panel3上放置四个TLabel控件，用于显示检测结果。最后在窗体上放置一个NMEcho、一个NMDayTime、一个NMTime和一个NMFTP控件。

双击Button1，创建此控件的OnClick事件处理函数，并加入如下所示的代码，用于检测主机是否提供echo、DayTime、Time和FTP服务。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 if(CheckBox1->Checked==true)
   {
    try
      {
       NMEcho1->Host = Edit1->Text;
       NMEcho1->TimeOut = StrToInt(Edit2->Text);
       NMEcho1->ReportLevel = Status_Basic;
       // 建立连接
       NMEcho1->Connect();
       NMEcho1->Disconnect();
      }
    catch(...)
      {
       Label1->Caption="主机不提供 echo 服务";
       NMEcho1->Disconnect();
      }
   }

 if(CheckBox2->Checked==true)
   {
    try
      {
       NMDayTime1->Host=Edit1->Text;
       NMDayTime1->TimeOut=StrToInt(Edit2->Text);
       AnsiString temp=NMDayTime1->DayTimeStr;
      }
    catch(...)
      {
       Label2->Caption="主机不提供 DayTime 服务";
```

```
        }
    }

  if(CheckBox3->Checked==true)
    {
     try
        {
         NMTime1->Host=Edit1->Text;
         NMTime1->TimeOut=StrToInt(Edit2->Text);
         AnsiString temp=NMTime1->TimeStr;
        }
     catch(...)
        {
         Label3->Caption="主机不提供 Time 服务";
        }
    }

  if(CheckBox4->Checked==true)
    {
     try
        {
         NMFTP1->Host=Edit1->Text;
         NMFTP1->TimeOut=StrToInt(Edit2->Text);
         NMFTP1->UserID="anonymous";
         NMFTP1->Password="user@263.net";
         NMFTP1->Connect();
         NMFTP1->Disconnect();
        }
     catch(...)
        {
         Label4->Caption="主机不提供 FTP 服务";
         NMFTP1->Disconnect();
        }
    }
}
//---------------------------------------------------------------------------
```

创建各个 NetMasters 控件的 OnConnect 事件处理函数，只有当特定端口支持某项协议（也即提供某项服务）时，才会触发 OnConnect 事件，代码如下所示：

```
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::NMEcho1Connect(TObject *Sender)
{
  Label1->Caption="主机提供 echo 服务";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1Connect(TObject *Sender)
{
 Label2->Caption="主机提供 DayTime 服务";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMTime1Connect(TObject *Sender)
{
 Label3->Caption="主机提供 Time 服务";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::NMFTP1Connect(TObject *Sender)
{
 Label4->Caption="主机提供 FTP 服务";
}
//---------------------------------------------------------------------------
```

编译并运行程序，填写主机地址然后单击“检测”按钮，就可以检测远程主机是否提供所选的服务。程序运行结果如图 1-1 所示。


图 1-1 程序运行结果


不要在 C++ Builder 的集成开发运行中运行此程序，而要脱离 C++ Builder 运

行此程序，才能顺利。因为，在 C++ Builder 的集成开发环境中，仍然会抛出异常。

### 专家点评

程序的核心是 Button1 的 OnClick 事件处理函数。事件处理函数的技巧是使用了 try 和 catch，在将异常传递给上一级前就将它处理掉。有关 NMEcho 控件的知识在 1.1 已有介绍，这里不再赘述。NMDayTime 控件是另一个非常有用的网络测试控件。NMDayTime 控件实现了 DayTime 协议。在客户与服务器建立了 TCP/IP 连接以后，调用该控件返回一个 ASCII 字符串。对于 TCP 和 UDP，默认的 DayTime 协议的端口都是 13。在程序代码中，首先将用户在编辑框控件中输入的数据拷贝给 NMDayTime 控件的 Host 和 Port，这些属性对于所有的标准 TCP/IP 协议都是必须的，然后可连接到 DayTime 服务器，并通过访问此控件的 DayTimeStr 属性来获取时间字符串。在这里有些读者可能会问，为什么没有像使用其他网络控件那样，首先调用控件的 Connect 方法与服务器建立连接。对于 NMDayTime 控件和 NMTime 控件来说，与服务器的连接是在使用了 DayTimeStr 和 TimeStr 属性的同时自动进行的。

最后，如果远程主机可以提供某项服务，当连接成功时就会触发控件的 OnConnect 事件。因此，在此事件处理函数中显示主机支持此项服务。

## 1.3　通过 Internet 发送邮件

### 关键所在

利用 NMSMTP 控件可以在程序中实现发送电子邮件的功能。

NMSMTP 控件实现了 SMTP。SMTP 是简单的邮件传输协议（Simple Mail Transfer Protocol）的缩写，这是服务器端 SMTP 电子邮件服务程序与客户机端电子邮件客户程序共同使用的协议之一，它是用于发送电子邮件的 Internet 网应用协议。此协议只支持 7 位编码 ASCII 文件的发送，在发送汉字、图形、图像等 8 位编码的二进制文件时，必须先进行 8 位到 7 位的代码转换。在接收方则需要进行逆转换。例如，流行的 UUENCODE.EXE 和 UUDECODE.EXE 命令文件就可以实现上述双向转换。使用 SMTP 协议可以通过网络发送邮件到其他的计算机上。NMSMTP 控件不但可以发送邮件，而且还可以校验正在接收的邮件。

### 实现与应用

选择菜单 File | New Application，创建一个新项目文件。在窗体 Form1 上先加入两个 TPanel 控件、一个 TMemo 控件和一个 TStatusBar 控件。将它们的 Align 属性分别设置为 alTop、alTop、alClient 和 alBottom，并将 StatusBar1 的 SimplePanel 属性设置为 true。在 Panel1 上放置四个 TLabel 控件、三个 TEdit 控件、一个 TListBox 控件和两个 TButton 控件。在 Panel2 上放置四个 TLabel 控件、四个 TEdit 控件、两个 TButton 控件和一个 TRadioGroup 控件。

最后再放置一个 TOpenDialog 控件和一个 NMSMTP 控件。可参看图 1-2 设置各个控件。

首先双击 Button1，创建它的 OnClick 事件处理函数，用于和 SMTP 服务器建立连接。代码如下所示：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 if(NMSMTP1->Connected)  //如果原先已经建立连接
   NMSMTP1->Disconnect();  //则断开连接

 NMSMTP1->Host=Edit4->Text;  //设置 SMTP 服务器地址
 NMSMTP1->UserID=Edit5->Text; //设置用户名
 NMSMTP1->Connect();  //建立连接
}
//---------------------------------------------------------------------------
```

双击 Button2，创建此控件的 OnClick 事件处理函数，并加入如下所示的代码，用于发送邮件。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 if (NMSMTP1->Connected)  //如果已经建立连接
     {
       NMSMTP1->SubType = mtPlain;
       switch(RadioGroup1->ItemIndex)
          {
           case 0:
             NMSMTP1->EncodeType = uuMime;
           case 1:
             NMSMTP1->EncodeType = uuCode;
          }

       NMSMTP1->PostMessage->FromAddress=Edit6->Text;
       NMSMTP1->PostMessage->FromName=Edit5->Text;
       NMSMTP1->PostMessage->ToAddress->Add(Edit1->Text);
       NMSMTP1->PostMessage->ToCarbonCopy->Add(Edit2->Text);
       NMSMTP1->PostMessage->ToBlindCarbonCopy->Add(Edit3->Text);
       NMSMTP1->PostMessage->Subject=Edit7->Text;
       NMSMTP1->PostMessage->Body->Assign(Memo1->Lines);
       NMSMTP1->PostMessage->Attachments->AddStrings(ListBox1->Items);
       NMSMTP1->SendMail();
```

```
    }
  else  //告诉用户先单击“连接”按钮和服务器建立连接之后再发送邮件
    ShowMessage("您必须首先单击“连接”按钮连接服务器.");
}
//---------------------------------------------------------------------------
```

双击 Button3 控件，创建它的 OnClick 事件处理函数，用于在邮件中插入附件，代码如下所示：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 if (OpenDialog1->Execute())
  ListBox1->Items->Add(OpenDialog1->FileName);//把附件的文件名加入到 ListBox1
}
//---------------------------------------------------------------------------
```

双击 Button4 控件，创建它的 OnClick 事件处理函数，删除原先想要插入的附件。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
 ListBox1->Items->Delete(ListBox1->ItemIndex);
}
//---------------------------------------------------------------------------
```

为了显示操作的成功与否，创建 NMSMTP1 控件的一些事件处理函数，在这些事件处理函数中通过状态条向用户显示信息，代码如下所示：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMSMTP1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="已和主机建立连接";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMSMTP1Failure(TObject *Sender)
{
 ShowMessage("操作失败");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMSMTP1Success(TObject *Sender)
{
 StatusBar1->SimpleText = "邮件已发送";
}
//---------------------------------------------------------------------------
```