# 네트워크 어플리케이션의 제작

# (Creating Network Application)

37 장에서는 소켓을 이용한 어플리케이션 제작 방법에 대해서 알아보았다. 이번 장에서는 델파이 4 에서 제공되는 NetMaster 의 컴포넌트 들을 소개하고, 이를 이용하여 작성할 수 있는 네트워크 어플리케이션을 소개한다.

## NetMaster 컴포넌트

델파이의 인터넷 컴포넌트는 2.01 버전의 NetManage 컴포넌트에서 시작되었다. 이 컴포넌트 들은 TCP/IP, SMTP, POP3, Telnet, FTP 등의 인터넷 서비스를 모두 지원했지만 그다지 높은 호응을 얻지는 못했는데, 그 이유는 델파이의 native VCL 컴포넌트가 아니라 액티브 X 컴포넌트였기 때문이다. 즉, 액티브 X 컨트롤을 같이 배포하고 이를 등록하지 않으면 사용할 수 없는 단점이 있었다. 그렇기 때문에 많은 개발자들이 프리웨어로 배포된 Francois Piette 등의 인터넷 컴포넌트를 사용하였다.
델파이 4 에서는 이러한 문제점을 시정하여 완전한 native VCL 컴포넌트로 변신한 NetMaster 컴포넌트 들을 만나볼 수 있다. NetMaster 컴포넌트에는 단순한 인터넷 표준 프로토콜을 지원하는 이외에 나름대로 커스터마이징이 가능한 원시 컴포넌트를 다수 지원하고 있기 때문에 수월하게 네트워크 프로그래밍을 할 수 있다. 그렇지만, 아쉬운 점이 있다면 가장 흔히 사용되는 HTML 컴포넌트는 여전히 액티브 X 컨트롤을 사용한다는 점이다. 그래도 델파이 3 에서의 HTML 컴포넌트가 버전 1.0 만 지원하여 무척 부족했지만, 델파이 4 에서는 버전 2.x 를 지원하기 때문에 대부분의 HTML 페이지를 표시할 수 있다.
그러면 NetMaster 컴포넌트 들의 종류와 이들에 대해 개략적으로 알아보도록 하자.

- TNMFTP 컴포넌트

TNMFTP 컴포넌트는 FTP 프로토콜을 이용하여 FTP 서버로 파일을 전송하거나, 파일을 다운로드 받기 위해서 사용된다. 이렇게 TNMFTP 컴포넌트를 이용하여 파일을 전송하기 위해서는 원격지 호스트와의 접속이 선행되어야 한다.
이를 위해서는 먼저 유효한 FTP 서버의 IP 주소를 Host 프로퍼티에 설정하고, Port 프로퍼티를 정확하게 설정해야 한다. 그리고 나서, 서버에서 허용하는 계정의 사용자 ID 와 패스워드를 UserID, Password 프로퍼티에 설정한다. 보통 대부분의 FTP 서버는 사용자 ID 로 Anonymous 와 패스워드로 E-mail 주소를 사용할 수 있도록 되어 있다. 이런 기본적인 프로퍼티 설정이 끝나면 Connect 메소드를 호출하여 서버에 접속한다.

이제 서버의 디렉토리 리스트를 얻어올 차례이다. 이를 위해서는 List 메소드를 호출하고, OnListItem 이벤트 핸들러에서 넘어오는 디렉토리 리스트의 아이템을 처리해 주어야 한다. 디렉토리를 변경하기 위해서는 ChangeDir 메소드를 사용한다.
원격지 호스트에 파일을 업로드하기 위해서는 업로드를 하기 위한 디렉토리에 먼저 위치한 뒤에 Upload 메소드를 호출하면 된다. 이 메소드에 로컬 컴퓨터의 파일의 이름과 원격지 허스트에 위치할 파일 이름을 파라미터로 지정할 수 있다.
파일을 다운로드할 때에는 먼저 List 메소드를 호출하여 다운로드할 파일을 찾을 수 있어야 한다. 그리고 나서 Download 메소드를 호출하는데, 여기에서 파라미터로 다운로드할 파일 이름과 로컬 드라이브에 저장할 파일의 경로아 이름을 지정한다.
디렉토리를 원격 호스트에 만들기 위해서는 MakeDirectory, 디렉토리를 지우기 위해서는 RemoveDir 메소드를 이용한다.

- TNMHTTP 컴포넌트

TNMHTTP 컴포넌트는 인터넷을 통해 HTTP 전송을 수행한다. URL 의 특성상 URL 에 호스트와 포트의 내용을 포함하기 때문에 Host 와 Port 프로퍼티는 설정할 필요가 없다.
문서를 Get 하고자 할 때에는 단순히 Get 메소드를 호출하면 된다. 이때 불러올 문서의 URL 을 지정하면 되는데, 이렇게 해서 불러온 데이터는 InputFileMode 프로퍼티의 내용에 따라 다르게 저장된다. 이 값이 True 이면 문서의 body 부분은 Body 프로퍼티에 지정된 파일에 저장되고, header 부분은 Header 프로퍼티에 지정된 파일에 저장된다. 반면에 이 값이 False 이면 Body 와 Header 프로퍼티에 직접 저장된다.
데이터를 Post 할 때에도 Post 메소드에 URL 을 첫번째 파라미터로 지정하고, 두번째 파라미터인 PostData 는 OutputFileMode 프로퍼티의 값에 따라 결정된다. OutputFileMode 프로퍼티의 값이 True 이면 PostData 파라미터에는 특정 위치에 전달될 데이터가 저장된 파일이 지정된다. 반면 이 값이 False 이면 전달된 데이터가 직접 PostData 파라미터에 지정된다. 이 메소드에 의해 전달되어 오는 문서를 저장하는 방법은 InputFileMode 에 의해 결정되는데, 그 내용은 Get 메소드와 동일하다.

- TNMNNTP 컴포넌트

TNMNNTP 컴포넌트는 인터넷 뉴스를 NNTP 프로토콜에 의해 인터넷 뉴스 서버에서 읽고, 메시지를 전달하기 위해 사용된다. TNMNNTP 컴포넌트의 핵심 함수를 이용할 때에는 먼저 뉴스 호스트에 접속해야 한다. 이를 위해서 Host 프로퍼티에 뉴스 서버의 주소를 설정하고, Connect 메소드를 이용하여 서버에 접속한다.
뉴스 그룹의 리스트를 얻기 위해서는 GetGroupList 메소드를 이용한다. 그리고, 기사를 읽어올 뉴스 그룹을 선택해야 하는데, SetGroup 메소드에 뉴스그룹의 이름을 파라미터로 넘

겨서 호출하면 된다. 인터넷 뉴스 기사를 읽을 때에는 일단 그룹을 선택한 뒤에 GetArticle 메소드를 호출하면 된다. 이렇게 하면 Body 프로퍼티와 Header 프로퍼티의 값을 이용해서 선택된 기사를 읽을 수 있게 된다.
새로운 뉴스 기사를 전송하기 위해서는 PostHeader, PostBody 프로퍼티에 메시지의 헤더와 내용을 채우고, PostArticle 메소드를 호출하면 된다.

- TNMPOP3 컴포넌트

TNMPOP3 컴포넌트는 인터넷 E-mail 을 POP3 서버로부터 가져오는 역할을 한다. 이를 위해서 먼저 Host 프로퍼티를 E-mail 서버의 주소로 설정하고, UserID 프로퍼티와 Password 프로퍼티에 사용자 이름과 패스워드를 지정하고 Connect 메소드를 호출하여 서버에 접속해야 한다.
인터넷 메일을 가져오기 위해서는 GetMailMessage 메소드를 호출하면 된다. 메일의 body 의 파트는 MailMessage 프로퍼티에 저장된다.

- TNMSMTP 컴포넌트

TNMSMTP 컴포넌트는 SMTP 프로토콜을 이용하여 인터넷 메일 서버에 메일을 전송하는 역할을 한다. 다른 컴포넌트와 마찬가지로 Host 와 Post 프로퍼티를 설정하고. Connect 메소드를 호출하여 메일 서버와 접속하고, Disconnect 메소드를 호출하여 접속을 중단한다.
메일을 보내기 위해서는 PostMessage 프로퍼티에 전송할 데이터를 지정하고, SendMail 메소드를 호출하면 된다. 경우에 따라서는 사용자가 연결된 호스트에 있는지 찾아볼 필요가 있는데 이럴 때에는 Verify 메소드를 이용한다. 그리고, 메일링 리스트의 멤버는 ExpandList 메소드를 이용하여 결정하는데, OnMailListReturn 이벤트 핸들러를 작성한다.

- TNMFinger

TNMFinger 컴포넌트는 인터넷 Finger 서버로부터 사용자에 대한 정보를 얻을 때 사용된다. 먼저 Finger 서버에 접속해야 하므로, Host 프로퍼티에 서버의 주소를 지정하고 Port 프로퍼티는 거의 대부분 79 번을 사용하므로 변경할 필요가 없다. 사용자에 대한 정보는 FingerStr 프로퍼티에서 얻을 수 있다.

- TNMUDP

TNMUDP 컴포넌트는 UDP 프로토콜을 이용하여 데이터 그램 패킷을 전송하는데 사용한다. 이 컴포넌트 역시 마찬가지로 패킷을 전송하기 전에 원격 호스트와 포트를 알아서

RemoteHost 와 RemotePort 프로퍼티에 값을 대입해야 한다.
실제로 데이터를 전송하기 위해서는 SendBuffer 또는 SendStream 메소드를 이용하는데 SendBuffer 메소드는 원격 호스트에 문자의 배열 형태로 데이터를 전송하며, SendStream 메소드는 데이터의 스트림을 전달한다.
UDP 데이터를 받을 때에는 LocalPort 프로퍼티를 반드시 설정해야 한다. 이 프로퍼티는 반드시 디자인 타임에서 설정해야 하며, 런타임에서 변경할 수 없다. 읽어올 UDP 데이터가 있으면 OnDataAvailable 이벤트가 호출되며 이 이벤트에서 ReadBuffer 메소드를 이용하여 데이터를 버퍼로 읽어들이거나 ReadStream 메소드를 이용하여 스트림으로 데이터를 읽는다.

- TNMDayTime

TNMDayTime 컴포넌트는 인터넷 daytime 서버로부터 RFC 867 에 정의된 날짜와 시간에 대한 정보를 불러온다. 사용방법은 Host 프로퍼티에 서버의 주소를 지정하고, 호스트에 접속하기만 하면 DayTimeStr 프로퍼티에 지정된 호스트의 날짜와 시간이 전송되어 저장되므로 이를 이용하기만 하면 된다.

- TNMTime

TNMTime 컴포넌트는 인터넷 time 서버에서 RFC 868 에 정의된 날짜와 시간에 대한 정보를 불러온다. 사용방법은 Host 프로퍼티에 서버의 주소를 지정하고, 호스트에 접속하기만 하면 TimeStr 프로퍼티에 지정된 호스트의 날짜와 시간이 전송되어 저장되므로 이를 이용하기만 하면 된다.

- TNMEcho

TNMEcho 컴포넌트는 텍스트를 인터넷 에코 서버에 전송하고, 이 내용을 다시 받을 때 사용되는데, RFC 862 에 정의된 프로토콜을 사용한다. 이를 이용해서 네트워크의 무결성과 속도를 측정한다.
서버와의 접속을 위해 Host 프로퍼티와 Port 프로퍼티를 설정하는데, 보통 에코 서버는 7 번 포트를 이용한다. 텍스트를 전송할 때에는 Echo 메소드를 이용한다.

- TNMUUProcessor

TNMUUProcessor 컴포넌트는 인터넷을 통해 전송되는 파일을 인코딩하고 디코딩할 때 사용된다. 사용 방법은 InputFile 프로퍼티를 처리할 파일로 설정하고, Method 프로퍼티에서

인코딩, 디코딩할 방법을 지정하고, OutputFile 프로퍼티에 결과로 생성될 파일을 지정한다. 그리고 실제로 실행을 위해서 Encode, Decode 메소드를 호출하면 된다.
사용하는 인코딩 메소드로는 MIME/Base 64 를 지원하는 uuMIME 과 UUEncoding/ Decoding 을 지원하는 uuCode 를 사용할 수 있다. 이 컴포넌트를 사용하는 예제를 다음 장에서 볼 수 있을 것이다.

- TNMURL

TNMURL 컴포넌트는 HTTP 전송에 대한 문자열과 URL 포맷 사이를 인코드, 디코드할 때 사용된다. 사용방법은 InputString 프로퍼티에 문자열이나 URL 포맷의 값을 설정하면 Encode 프로퍼티에 URL 포맷으로 인코드된 문자열을 포함하게 되고, Decode 프로퍼티에는 디코딩된 문자열이 포함된다. 이때 에러가 발생하면 OnError 이벤트가 호출된다.

- TNMMsg

TNMMsg 컴포넌트는 간단 메시지를 전송하기 위해서 사용하는데, TNMMsgServ 컴포넌트와 함께 사용하여야 한다. 메시지를 전송하기 전에 Host 프로퍼티와 Port 프로퍼티에 해당되는 메시지 서버의 IP 주소와 포트를 설정하고, FromName 프로퍼티에서 메시지를 전송한 사람에 대한 정보를 설정하고, PostIt 메소드를 호출하면 메시지가 전송된다.

- TNMMsgServ

TNMMsgServ 컴포넌트는 TNMMsg 컴포넌트에서 전송된 메시지를 받는 역할을 한다. 어플리케이션을 디자인할 때 Port 프로퍼티에는 메시지 서버가 사용할 포트를 지정하는데, 디폴트 값을 사용해도 무방하다. 메시지 서버에 전송된 메시지를 받아서 처리하기 위해서는 OnMsg 이벤트 핸들러를 작성하면 된다,

- TNMStrm

TNMStrm 컴포넌트는 스트림을 스트림 서버에 전송하기 위해서 TNMStrmServ 컴포넌트와 함께 사용된다. 먼저 Host 와 Port 프로퍼티를 서버에 맞도록 설정하고, FromName 프로퍼티에 스트림을 전송하는 사람의 정보를 지정하고, PostIt 메소드를 이용하여 스트림을 전송하면 된다.

- TNMStrmServ

TNMStrmServ 컴포넌트는 TNMStrm 컴포넌트에서 전송하는 스트림을 처리하는 역할을 한다. TNMMsgServ 컴포넌트와 마찬가지로 Port 프로퍼티를 설정하고, OnMsg 이벤트에서 전송된 스트림을 처리하면 된다.

- TPowersock

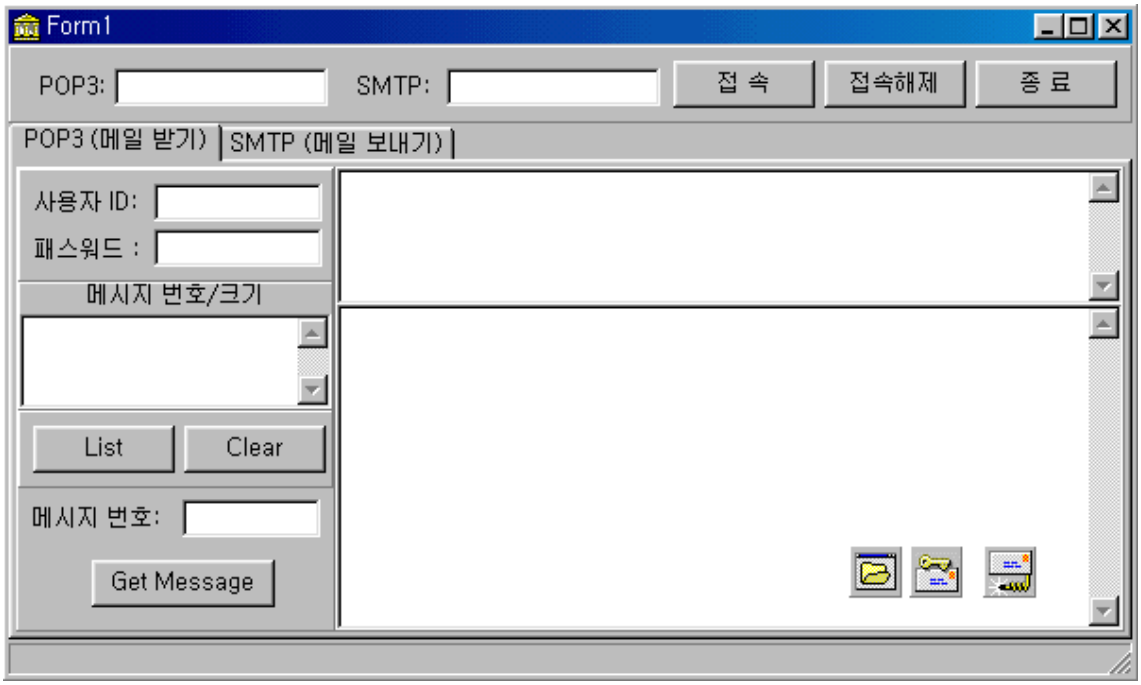
TPowersock 컴포넌트는 TCP 통신을 이용하여 다양한 인터넷 프로토콜을 이용할 수 있도록 제공하는 기초 클래스이다. 그러므로 이 컴포넌트는 상속을 받아서 새로이 구성해야 하는 것으로, Connect 메소드를 호출하면 Host 와 Port 프로퍼티에 지정한 원격 호스트의 서비스에 접속하게 된다. 원격 호스트에서 받아온 데이터는 Read, ReadLn, CaptureFile, CaptureStream, CaptureString 메소드를 이용하여 읽을 수 있다. 그리고, 데이터를 원격 호스트에 전송할 때에는 Write, WriteLn, SendFile, SendStream 메소드를 이용한다. 그리고, Transaction 메소드는 원격 호스트에 데이터를 전송하고 서버로부터 응답을 받는다.

- TNMGeneralServ

TNMGeneralServer 컴포넌트는 다중 쓰레드를 지원하는 인터넷 서버를 개발할 때 RFC 표준을 지원하는 서버나 사용자 정의 서버를 개발할 때 사용되는 기초 클래스이다.
TNMGeneralServer 클래스를 상속하는 서버 클래스를 제작하기 위해서는 Server 메소드를 오버라이드해야 하는데, 여기에서 서버가 클라이언트 연결에 어떻게 반응할 것인지를 결정한다. 클라이언트와의 상호 작용은 TPowersock 의 read, write 메소드를 이용한다.
주의할 점은 Port 프로퍼티를 디자인 타임에서 설정해야 한다는 것이다.

## 메일 클라이언트 어플리케이션의 제작

그러면, 표준 인터넷 프로토콜을 지원하는 컴포넌트 중에서 POP3 와 SMTP 프로토콜을 이용한 메일 클라이언트를 제작해보도록 하자.
먼저 폼을 디자인해야 하는데, 나름대로 성의를 가지고 여러 개의 패널 객체와 Align 프로퍼티를 적절히 설정해서 깔끔한 인터페이스를 구성해보도록 하자. POP3 와 SMTP 클라이언트를 하나의 폼으로 구성하되, 메일을 받는 인터페이스와 보내는 인터페이스는 TPageControl 컴포넌트를 이용하여 다른 페이지로 구성한다.
메일을 보낼 때에는 파일을 Attach 하여 보낼 수 있도록 TOpenDialog 컴포넌트를 추가한다. 그리고, 실제 프로토콜을 관리하는 TNMPOP3, TNMSMTP 컴포넌트를 폼에 추가하도록 하자.
필자가 구성한 폼의 디자인은 다음과 같다. 이와 비슷하게 필요한 형태로 폼을 디자인하면 될 것이다.




'접 속' 버튼을 클릭하면 POP3 와 SMTP 라벨 옆에 있는 에디트 박스의 내용을 바탕으로 서버에 접속한다. POP3 의 경우에는 사용자 ID 와 패스워드를 이용하여 메일 서버에 접속하는 과정이 필요하다. 이때 패스워드는 입력할 때 입력하는 문자가 보이지 않거나, 특수문자로 처리하는데 대개의 경우 '*' 문자를 이용한다. 이럴 때에는 TMaskEdit 컴포넌트를 이용하는 것이 좋다. MaskEdit1 객체의 PasswordChar 프로퍼티를 '*'로 설정한다. 이 버튼의 OnClick 이벤트 핸들러를 다음과 같이 작성하여, 메일 서버에 접속하도록 한다.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  NMPOP31.AttachFilePath := '.';
  NMPOP31.DeleteOnRead := FALSE;
  NMPOP31.ReportLevel := Status_Basic;
  NMPOP31.TimeOut := 20000;
  NMPOP31.Host := Edit1.Text;
  NMPOP31.UserID := Edit3.Text;
  NMPOP31.Password := MaskEdit1.Text;
  NMPOP31.Connect;
  NMSMTP1.TimeOut := 20000;
  NMSMTP1.ReportLevel := Status_Basic;
  NMSMTP1.Host := Edit2.Text;
```

```
  NMSMTP1.UserID := Edit5.Text;
  NMSMTP1.Connect;
end;
```

접속을 하기 위해서는 TNMPOP3, TNMSMTP1 컴포넌트 모두 Connect 메소드를 호출하면 된다. 이때 Host 프로퍼티에 메일 서버에 IP 주소를 대입하고, TimeOut 프로퍼티는 접속을 시도할 시간을 msec 단위로 설정한다. ReportLevel 프로퍼티는 OnStatus 이벤트에서 상태 메시지를 어떻게 표시할 것인지를 결정하는 프로퍼티인데 보통은 Status_Basic 으로 설정한다. POP3 컴포넌트에서의 DeleteOnRead 프로퍼티는 메시지를 읽을 경우에 메일 서버에서 읽은 메시지를 제거할 것인지 여부를 결정한다. 이 값을 True 로 할 경우에는 메시지를 클라이언트로 다운로드하면 서버에서 메시지가 삭제되며, False 로 설정할 경우에는 메일 클라이언트에서 읽더라도 서버에 메시지가 그대로 남게 된다. 이 경우에는 메시지를 직접 삭제할 수 있도록 해야 한다.
마찬가지로 '접속해제' 버튼을 클릭하면 Disconnect 메소드를 호출하면 된다.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  NMPOP31.Disconnect;
  NMSMTP1.Disconnect;
end;
```

POP3 페이지에서 'List' 버튼을 클릭하면 POP3 메일 서버에 도착한 메시지를 나열하게 되는데, 이렇게 나열되는 메시지는 OnList 이벤트 핸들러에서 처리할 수 있다. 'Clear' 버튼을 클릭하면 메모 컴포넌트의 내용을 지우도록 한다.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  NMPOP31.List;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Memo3.Lines.Clear;
end;
```

OnList 이벤트 핸들러는 다음과 같이 작성하여, 전달된 메시지의 번호와 크기를 메모 컴포

넌트에 나열하도록 한다.

```
procedure TForm1.NMPOP31List(Msg, Size: Integer);
begin
  Memo3.Lines.Add(IntToStr(Msg) + ' / ' + IntToStr(Size));
end;
```

이렇게 메모 컴포넌트에 나열된 메일 서버의 메시지 중에서 읽어올 메시지 번호를 에디트 박스에 지정하고, 'Get Message' 버튼을 클릭하면 해당되는 메시지를 읽어와서 header 는 우측의 위쪽 메모 컴포넌트에, body 는 아래쪽 메모 컴포넌트에 표시한다.
'GetMessage' 버튼의 OnClick 이벤트 핸들러는 다음과 같이 작성한다.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  NMPOP31.GetMailMessage(StrToInt(Edit4.Text));
  Memo1.Lines.Assign(NMPOP31.MailMessage.Head);
  Memo2.Lines.Assign(NMPOP31.MailMessage.Body);
  if NMPOP31.MailMessage.Attachments.Text <> '' then
    ShowMessage('Attachments: ' + #10#13 + NMPOP31.MailMessage.Attachments.Text);
end;
```

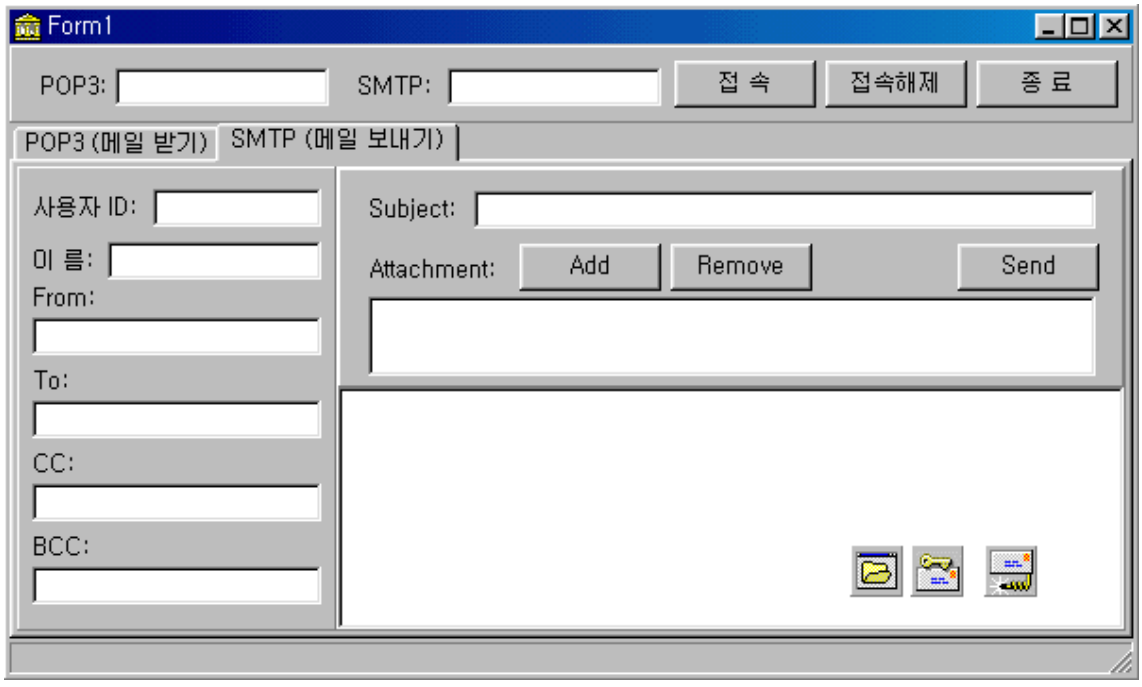
즉, 메시지를 읽어올 때에는 GetMailMessage 메소드에서 파라미터에 넘겨진 번호의 메시지를 읽어오게 된다. 이렇게 읽어온 메시지는 MailMessage 프로퍼티에 저장되는데, 각각 header 와 body 부분으로 나눌 수 있다. 이들은 모두 TStrings 데이터 형이기 때문에 메모 컴포넌트에 직접 대입할 수 있다. MailMessage 의 Attachment 프로퍼티에는 attach 된 파일을 나타내는데, 파일의 이름을 Text 프로퍼티를 이용하여 얻을 수 있다.
이것으로 버튼의 이벤트 핸들러는 모두 작성하였다. 이제는 TNMPOP3 컴포넌트의 이벤트 핸들러를 작성하도록 하자. 대개의 경우 TStatusBar 컴포넌트에 메일이 전달되는 상황에 대한 정보를 표시하는 정도로 작성한다.

```
procedure TForm1.NMPOP31Connect(Sender: TObject);
begin
  StatusBar1.SimpleText := 'POP3 Server Connected !';
end;

procedure TForm1.NMPOP31Disconnect(Sender: TObject);
```

```
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := 'POP3 Server Disconnected !';
end;

procedure TForm1.NMPOP31Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := 'POP3: ' + Status;
end;

procedure TForm1.NMPOP31ConnectionFailed(Sender: TObject);
begin
  ShowMessage('POP3 Server Connection Failed');
end;

procedure TForm1.NMPOP31Failure(Sender: TObject);
begin
  ShowMessage('작업이 실패했습니다.');
end;

procedure TForm1.NMPOP31HostResolved(Sender: TComponent);
begin
  StatusBar1.SimpleText := 'POP3 Host Resolved';
end;

procedure TForm1.NMPOP31Success(Sender: TObject);
begin
  StatusBar1.SimpleText := 'POP3 Operation Successful !';
end;

procedure TForm1.NMPOP31PacketRecvd(Sender: TObject);
begin
  StatusBar1.SimpleText := IntToStr(NMPOP31.BytesRecvd) + ' Bytes of ' +
    IntToStr(NMPOP31.BytesTotal) + ' Received';
end;
```

```
procedure TForm1.NMPOP31RetrieveStart(Sender: TObject);
begin
  Form1.Cursor := crHourGlass;
  StatusBar1.SimpleText := 'Retrieval Start';
end;

procedure TForm1.NMPOP31RetrieveEnd(Sender: TObject);
begin
  Form1.Cursor := crDefault;
  StatusBar1.SimpleText := 'Retrieval End';
end;
```

이것으로 POP3 에 대한 부분은 모두 완성되었다.
이번에는 SMTP 클라이언트 기능을 추가할 차례이다. 인터페이스 디자인은 다음과 같다.




SMTP 클라이언트에서는 ‘Add’, ‘Remove’ 버튼을 클릭하여 attach 될 파일을 파일열기 대화상자를 이용하여 리스트 박스에 추가하도록 ‘Add’, ‘Remove’ 버튼의 OnClick 이벤트 핸들러를 다음과 같이 작성한다.

```
procedure TForm1.Button6Click(Sender: TObject);
```

```
begin
  If OpenDialog1.Execute then
    ListBox1.Items.Add(OpenDialog1.FileName);
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  ListBox1.Items.Delete(ListBox1.ItemIndex);
end;
```

이어서 'Send' 버튼을 클릭하면 이들과 함께 메모 컴포넌트의 내용을 To, CC, BCC 로 지정된 주소에 전송한다. 이때 보내는 사람에 대한 정보를 인터넷 메일 주소와 실제 이름과 함께 전송한다. 'Send' 버튼의 OnClick 이벤트 핸들러는 다음과 같이 작성한다.

```
procedure TForm1.Button8Click(Sender: TObject);
begin
  NMSMTP1.PostMessage.FromAddress := Edit7.Text;
  NMSMTP1.PostMessage.FromName := Edit6.Text;
  NMSMTP1.PostMessage.Subject := Edit11.Text;
  NMSMTP1.PostMessage.ToAddress.Add(Edit8.Text);
  NMSMTP1.PostMessage.ToBlindCarbonCopy.Add(Edit10.Text);
  NMSMTP1.PostMessage.ToCarbonCopy.Add(Edit9.Text);
  NMSMTP1.PostMessage.Attachments.AddStrings(Listbox1.Items);
  NMSMTP1.PostMessage.Body.Assign(Memo4.Lines);
  NMSMTP1.SendMail;
end;
```

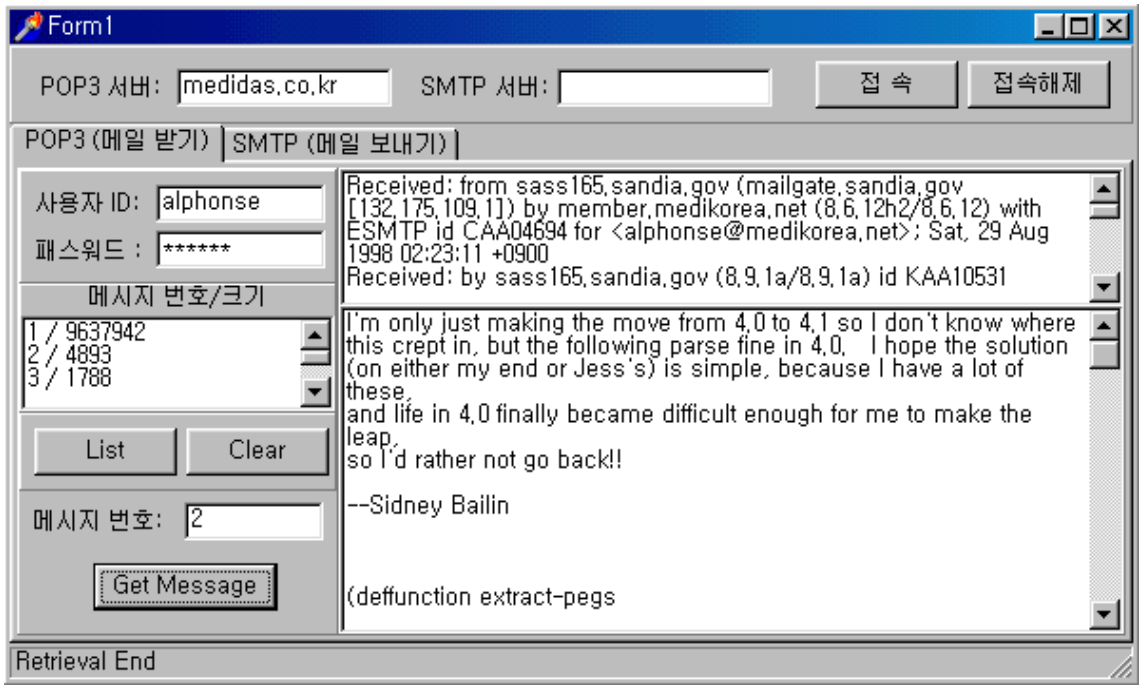
POP3 와 마찬가지로 SMTP 컴포넌트에서도 여러가지 이벤트에 대한 상황을 표시할 수 있도록 다음과 같이 이벤트 핸들러들을 작성한다.

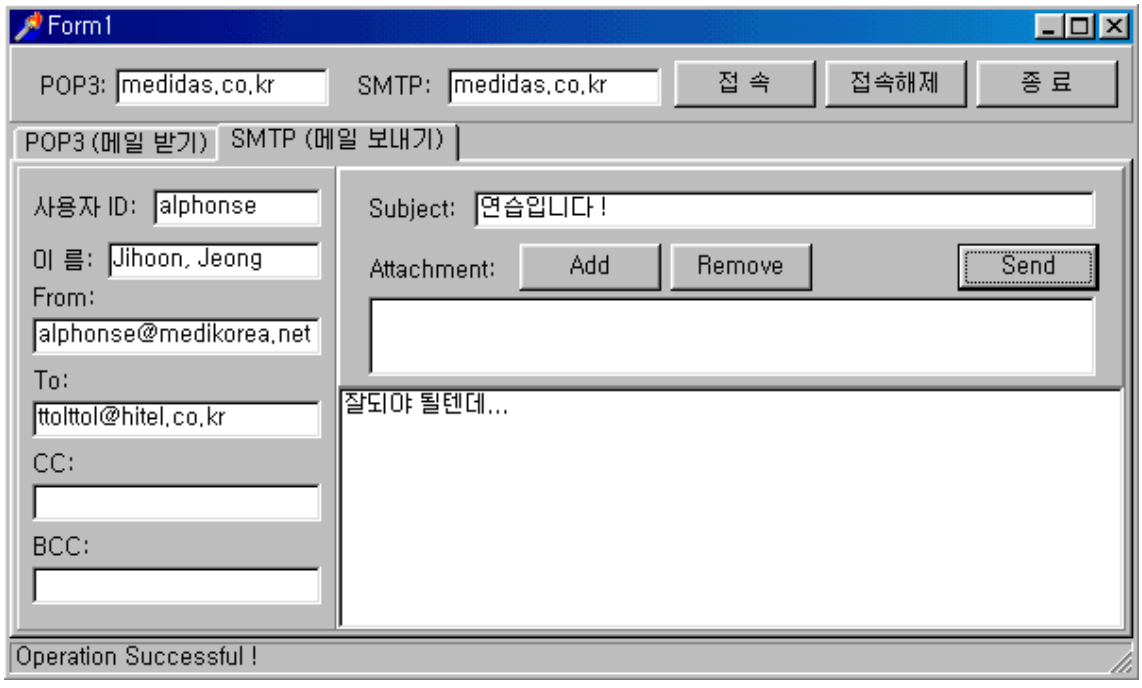
```
procedure TForm1.NMSMTP1Connect(Sender: TObject);
begin
  StatusBar1.SimpleText := 'SMTP Server Connected !';
end;

procedure TForm1.NMSMTP1Disconnect(Sender: TObject);
```

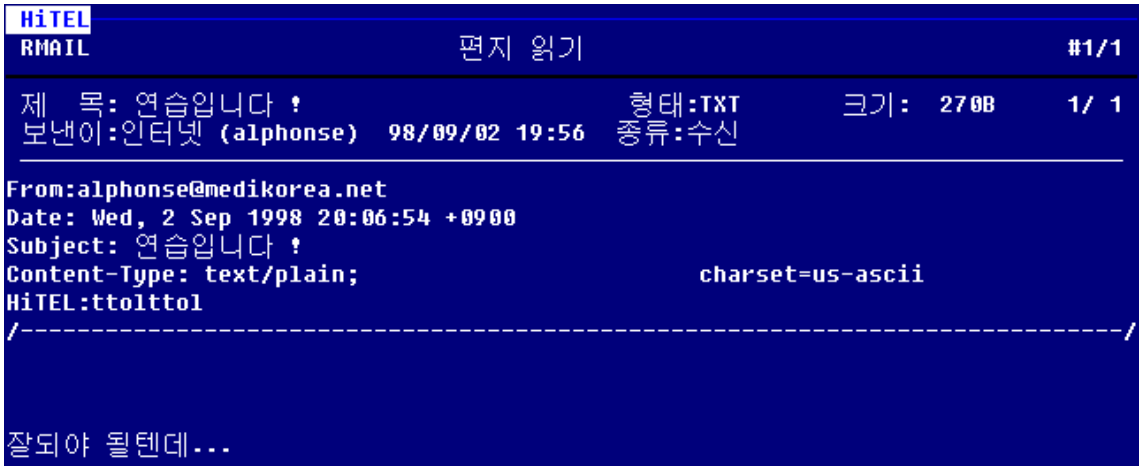
```
begin
  If StatusBar1 <> nil then
    StatusBar1.SimpleText := 'SMTP Server Disconnected';
end;

procedure TForm1.NMSMTP1Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := 'SMTP: ' + Status;
end;

procedure TForm1.NMSMTP1EncodeStart(Filename: String);
begin
  StatusBar1.SimpleText := 'Encoding ' + Filename;
end;

procedure TForm1.NMSMTP1EncodeEnd(Filename: String);
begin
  StatusBar1.SimpleText := 'Finished Encoding ' + Filename;
end;

procedure TForm1.NMSMTP1ConnectionFailed(Sender: TObject);
begin
  ShowMessage('SMTP Server Connection Failed');
end;

procedure TForm1.NMSMTP1Failure(Sender: TObject);
begin
  StatusBar1.SimpleText := 'SMTP Operation Failed';
end;

procedure TForm1.NMSMTP1HostResolved(Sender: TComponent);
begin
  StatusBar1.SimpleText := 'SMTP Host Resolved';
end;
```

```
procedure TForm1.NMSMTP1PacketSent(Sender: TObject);
begin
  StatusBar1.SimpleText := IntToStr(NMSMTP1.BytesSent) + ' Bytes of ' +
    IntToStr(NMSMTP1.BytesTotal) + ' Sent';
end;

procedure TForm1.NMSMTP1RecipientNotFound(Recipient: String);
begin
  ShowMessage('Recipient "' + Recipient + '" not found !');
end;

procedure TForm1.NMSMTP1SendStart(Sender: TObject);
begin
  StatusBar1.SimpleText := 'Sending Message ...';
end;

procedure TForm1.NMSMTP1Success(Sender: TObject);
begin
  StatusBar1.SimpleText := 'Operation Successful !';
end;
```

이것으로 그럴 듯한 인터넷 메일 클라이언트가 작성되었다. 다음 그림들은 필자가 메일 클라이언트를 사용하여 메일을 받고, 메시지를 작성하여 하이텔의 필자 ID 로 전송한 뒤에 이를 하이텔에 접속하여 메일이 도착했는지 확인하는 그림이다.


Form1
POP3 서버: medidas.co.kr
SMTP 서버:
접 속
접속해제
POP3 (메일 받기)
SMTP (메일 보내기)
사용자 ID: alphonse
패스워드 : ******
메시지 번호/크기
1 / 9637942
2 / 4893
3 / 1788
List
Clear
메시지 번호: 2
Get Message
Received: from sass165.sandia.gov (mailgate.sandia.gov [132.175.109.1]) by member.medikorea.net (8.6.12h2/8.6.12) with ESMTP id CAA04694 for <alphonse@medikorea.net>; Sat, 29 Aug 1998 02:23:11 +0900
Received: by sass165.sandia.gov (8.9.1a/8.9.1a) id KAA10531
I'm only just making the move from 4.0 to 4.1 so I don't know where this crept in, but the following parse fine in 4.0, I hope the solution (on either my end or Jess's) is simple, because I have a lot of these,
and life in 4.0 finally became difficult enough for me to make the leap,
so I'd rather not go back!!
--Sidney Bailin
(deffunction extract-pegs
Retrieval End

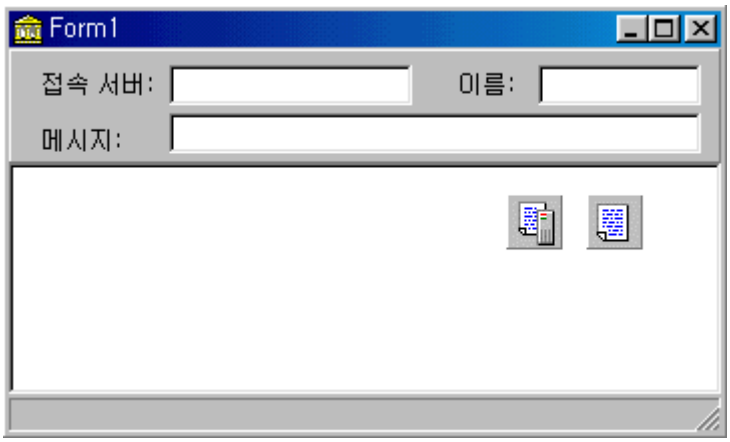

Form1
POP3: medidas.co.kr
SMTP: medidas.co.kr
접 속
접속해제
종 료
POP3 (메일 받기)
SMTP (메일 보내기)
사용자 ID: alphonse
이 름: Jihoon, Jeong
From:
alphonse@medikorea.net
To:
ttolttol@hitel.co.kr
CC:
BCC:
Subject: 연습입니다 !
Attachment:
Add
Remove
Send
잘되야 될텐데...
Operation Successful !




## 메시지 클라이언트/서버 어플리케이션의 제작

이번에는 NetMaster 의 메시지 클라이언트와 서버 컴포넌트를 이용하여 간단한 메시지를 주고받을 수 있는 메시지 클라이언트/서버 어플리케이션을 만들어 보자. 컴포넌트의 사용 방법을 익히고자 하는 것이 목적이므로 1:1 채팅 어플리케이션과 마찬가지로 하나의 어플리케이션이 서버이면서 동시에 클라이언트 역할을 하도록 만들도록 한다.

먼저 폼에 패널을 하나 얹고 패널 위에 메시지 서버의 IP 주소를 지정할 에디트 박스와 보내는 사람의 이름을 설정할 에디트 박스, 그리고 실제 메시지를 입력할 에디트 박스를 추가한다. 그리고, 메모 컴포넌트를 폼에 추가하여 전달되는 메시지를 표시하도록 한다. 그리고 다음과 같이 TStatusBar 컴포넌트를 추가하여 인터넷 메일 클라이언트에서와 마찬가지로 메시지 컴포넌트의 상태를 표시할 수 있도록 한다.




소켓 프로그래밍을 이용한 채팅 프로그램보다 사용방법은 더 간단하다. 다음과 같이 서버의 IP 주소와 보내는 사람의 이름을 각각 Host, FromName 프로퍼티에 지정하고, 보내는 메시지 내용을 PostIt 메소드를 이용하여 전송하는 것으로 끝이다.

```
procedure TForm1.Edit3KeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    NMMsg1.Host := Edit1.Text;
    NMMsg1.FromName := Edit2.Text;
    NMMsg1.PostIt(Edit3.Text);
  end;
end;
```

이제는 메시지 서버 컴포넌트와 메시지 컴포넌트의 각 이벤트를 처리하기만 하면 된다. 먼저 메시지 서버 컴포넌트의 OnClientContact 이벤트는 클라이언트가 접속했을 때 발생하게 된다.

```
procedure TForm1.NMMSGServ1ClientContact(Sender: TObject);
begin
  NMMsgServ1.ReportLevel := Status_Basic;
  NMMsgServ1.TimeOut := 90000;
  StatusBar1.SimpleText := 'Client connected';
end;
```

그리고, 전송된 메시지는 메시지 서버 컴포넌트의 OnMSG 이벤트에서 처리할 수 있다.

```
procedure TForm1.NMMSGServ1MSG(Sender: TComponent; const sFrom,
  sMsg: String);
begin
  Memo1.Lines.Add('(' + sFrom + ') ' + sMsg);
end;
```

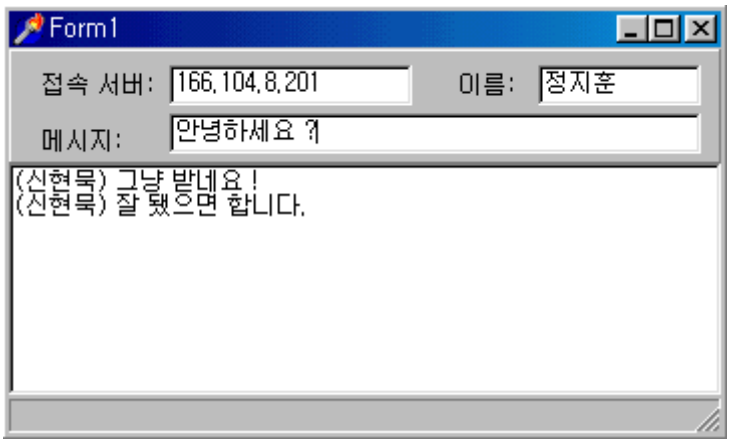
메시지 서버 컴포넌트의 상태에 대한 정보는 OnStatus 이벤트 핸들러에서 처리한다.

```
procedure TForm1.NMMSGServ1Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := Status;
end;
```

그 다음에는 메시지 클라이언트 컴포넌트의 이벤트를 처리해야 하는데, 이들의 처리 방법은 앞서 설명한 인터넷 메일 컴포넌트 들과 큰 차이가 없으므로 자세한 설명은 생략하도록 하겠다.

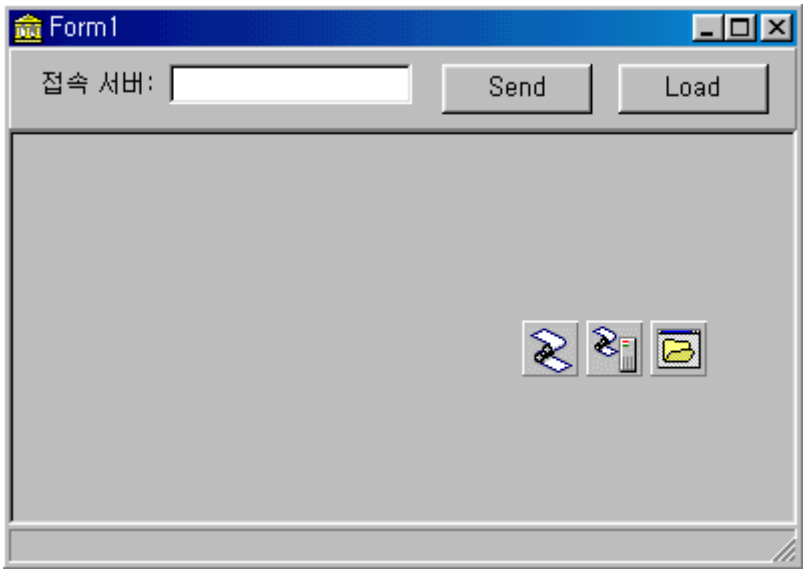
```
procedure TForm1.NMMsg1Connect(Sender: TObject);
begin
  StatusBar1.SimpleText := 'Connected';
end;

procedure TForm1.NMMsg1ConnectionFailed(Sender: TObject);
begin
  ShowMessage('Connection Failed');
end;

procedure TForm1.NMMsg1Disconnect(Sender: TObject);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := 'Disconnected';
end;

procedure TForm1.NMMsg1HostResolved(Sender: TComponent);
begin
  StatusBar1.SimpleText := 'Host Resolved';
end;

procedure TForm1.NMMsg1MessageSent(Sender: TObject);
begin
  ShowMessage('Message sent!');
end;

procedure TForm1.NMMsg1Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := Status;
end;
```

이제 이 프로그램을 컴파일하고 실행한 뒤에 1:1 채팅을 하듯이 사용해보자. 다음 그림과 같이 무리없이 잘 동작할 것이다.




## 스트림 클라이언트/서버 어플리케이션의 제작

이번에는 스트림 서버와 클라이언트 컴포넌트를 이용하여 파일을 전송하고, 이 파일의 내용을 볼 수 있는 어플리케이션을 만들어 보자. 먼저 서버의 IP 주소를 지정할 수 있는 에디트 박스와 버튼 2 개와 TStatusBar 컴포넌트, 그리고 TOleContainer 컴포넌트와 TOpenDialog, 스트림 클라이언트와 서버 컴포넌트를 각각 하나씩 다음과 같이 폼에 추가하도록 한다.




'Send' 버튼을 클릭하면 파일열기 대화상자에서 지정한 파일을 전송하며, 'Load' 버튼을 클

릭하면 이 파일을 OLE 컨테이너에서 볼 수 있도록 해준다.
먼저 'Send' 버튼의 OnClick 이벤트 핸들러를 다음과 같이 작성한다.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  MyFStream: TFileStream;
begin
  If OpenDialog1.Execute then
  begin
    NMStrm1.Host := Edit1.Text;
    NMStrm1.FromName := ExtractFileName(OpenDialog1.FileName);
    MyFStream := TFileStream.Create(OpenDialog1.FileName, fmOpenRead);
    try
      NMStrm1.PostIt(MyFStream);
    finally
      MyFStream.Free;
    end;
  end;
end;
```

사용 방법은 Host 프로퍼티에 접속할 서버의 IP 주소를 대입하고, 지정된 파일 스트림을 PostIt 메소드를 이용하여 서버로 전송한다. 참고로 FromName 프로퍼티에 파일 이름을 지정하면 파일의 확장자를 알 수 있으므로 이를 활용한다. 즉, 스트림에 대한 정보를 FromName 프로퍼티를 이용하여 문자열로 전송하는 것이다.
이렇게 전송된 파일 스트림은 스트림 서버 컴포넌트의 OnMSG 이벤트에서 받아볼 수 있는데, 이 이벤트 핸들러에서 임시 파일로 저장할 수 있도록 한다.
스트림 서버 컴포넌트의 OnMSG 이벤트 핸들러를 다음과 같이 작성한다.

```
procedure TForm1.NMStrmServ1MSG(Sender: TComponent; const sFrom: String;
  strm: TStream);
var
  MyFStream: TFileStream;
begin
  Extension := ExtractFileExt(sFrom);
  if FileExists(ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension) then
    DeleteFile(ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension);
```

```
  MyFStream
    := TFileStream.Create(
      ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension, fmCreate);
  try
    MyFStream.CopyFrom(strm, strm.Size);
  finally
    MyFStream.Free;
  end;
end;
```

이 이벤트 핸들러에서 파일 이름이 'tmp'이고 파일 확장자는 앞서 스트림 클라이언트 컴포넌트에서 FromName 프로퍼티에 지정한 문자열이 sFrom 파라미터로 넘어오므로, 이 값을 이용하여 확장자를 지정할 수 있다. 여기서는 디렉토리를 현재 어플리케이션이 실행되고 있는 디렉토리에 저장하도록 하였다.
이렇게 저장된 파일을 OLE 컨테이너에서 보여주는 역할을 하는, 'Load' 버튼은 비교적 구현하기 쉽다. 파일이름이 'tmp'이고, 지정된 확장자를 가진 파일을 CreateObjectFromFile 메소드를 이용해서 OLE 컨테이너에 보여준다.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if FileExists(ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension) then
    OleContainer1.CreateObjectFromFile(ExtractFilePath(ParamStr(0)) + '₩tmp'
    + Extension, False);
end;
```

그리고, 폼의 OnClose 이벤트 핸들러에서 이렇게 생성된 임시 파일을 삭제하도록 한다.

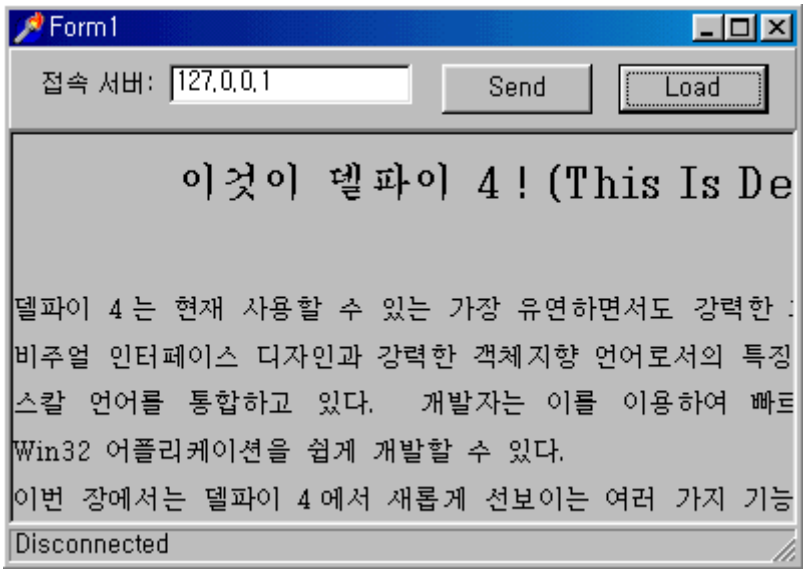
```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if FileExists(ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension) then
    DeleteFile(ExtractFilePath(ParamStr(0)) + '₩tmp' + Extension);
end;
```

스트림 컴포넌트의 다른 이벤트 핸들러는 메시지 클라이언트/서버와 마찬가지로 상태 정보를 보여주는 역할을 한다. 각각의 이벤트 핸들러는 다음과 같이 구현하며, 그다지 어려운 내용이 아니므로 자세한 설명은 생략하도록 하겠다.

```
procedure TForm1.NMStrm1Connect(Sender: TObject);
begin
  StatusBar1.SimpleText := 'Connected';
end;

procedure TForm1.NMStrm1Disconnect(Sender: TObject);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := 'Disconnected';
end;

procedure TForm1.NMStrm1HostResolved(Sender: TComponent);
begin
  StatusBar1.SimpleText := 'Host Resolved';
end;

procedure TForm1.NMStrm1Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := Status;
end;

procedure TForm1.NMStrmServ1ClientContact(Sender: TObject);
begin
  NMStrmServ1.ReportLevel := Status_Basic;
  NMStrmServ1.TimeOut := 90000;
  StatusBar1.SimpleText := 'Client connected';
end;

procedure TForm1.NMStrmServ1Status(Sender: TComponent; Status: String);
begin
  if StatusBar1 <> nil then
    StatusBar1.SimpleText := Status;
end;
```

```
procedure TForm1.NMStrm1ConnectionFailed(Sender: TObject);
begin
  ShowMessage('Connection Failed');
end;

procedure TForm1.NMStrm1MessageSent(Sender: TObject);
begin
  ShowMessage('Stream Sent');
end;

procedure TForm1.NMStrm1PacketSent(Sender: TObject);
begin
  StatusBar1.SimpleText := IntToStr(NMStrm1.BytesSent) + ' of '
   + IntToStr(NMStrm1.BytesTotal) + ' Sent';
end;
```

이것으로 스트림 서버/클라이언트 어플리케이션이 완성되었다. 프로그램을 컴파일하고 실행한 뒤에 Send 버튼을 클릭하여 파일을 지정하면, 서버로 파일 스트림이 전송될 것이다. 다음 그림은 이 책의 1 장에 해당되는 워드 파일을 보여주는 것이다.




## 정 리 (Summary)

델파이는 2.01 버전부터 지원하던 액티브 X 컨트롤 형태의 인터넷 컴포넌트를 버리고, 델파

이 4 부터는 native VCL 형태의 뛰어난 컴포넌트를 제공하여 지금까지의 많은 불평들을 해소시켜 주었다. 또한 TPowersock, TGeneralServer 와 같은 컴포넌트를 이용하면 RFC 에서 제정하는 새로운 프로토콜을 지원하는 컴포넌트를 직접 만들거나, 프로토콜부터 새로 만들어서 사용할 수도 있다.
그렇지만, 중요한 것은 이러한 과정을 제대로 이해하기 위해서는 윈속 API 를 이용한 실제 구현 방법에 대해서 알아야 할 것이다. NetMaster 컴포넌트에 대해서 아쉬운 것은 델파이 4 에 번들되어 있지만, 소스 코드는 공개되지 않았다는 점이다. 이런 측면에서 Francois Piette 의 ICS 컴포넌트 suite 를 추천하고 싶다. 이 컴포넌트는 델파이 수퍼 페이지나 델파이 델리와 같은 사이트에서 쉽게 구할 수 있다.