## Лабораторная работа №1

**СОЗДАНИЕ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ В DELPHI НА ПРИМЕРЕ СЕТЕВОГО ЧАТА**

**Цель работы:** ознакомиться с понятием socket*'*ов и с их помощью реализовать сетевой чат в среде Delphi.

## Программирование socket'ов в Delphi

При работе с сетью можно использовать возможности, которые предоставляет среда разработки Delphi. Объекты упрощают программирование и скрывают некоторые особенности реализации протоколов и сети.

Socket (гнездо, разъем) – абстрактное программное понятие, используемое для обозначения в прикладной программе конечной точки канала связи с коммуникационной средой, образованной вычислительной сетью. При использовании протоколов TCP/IP можно говорить, что socket является средством подключения прикладной программы к порту локального узла сети. С помощью socket легко реализовать большинство известных протоколов, которые используются каждый день при выходе в Интернет. Достаточно только назвать HTTP, FTP, POP3, SMTP и т.д.

## Сетевые порты

Когда программа соединяется с сервером, то она открывает на вашем компьютере какой-нибудь сетевой порт и сообщает серверу, что именно с этим портом она работает. После этого сервер будет посылать на ваш компьютер пакеты данных, в которых будет указан сетевой адрес компьютера и номер порта. По IP-адресу пакет будет доставлен до вашего компьютера, а по номеру порта операционная система определит, что именно для вашей программы предназначается пришедший пакет.

Для соединения с сервером вам надо знать не только IP-адрес сервера, но и порт, на котором работает программа, потому что на сервере может работать множество сетевых программ, и все они используют свои порты.

Только одна программа может открыть определенный порт. Если бы две программы могли открывать, например, 21-й порт, то Windows (или любая другая операционная система) уже не смогла бы определить, какой из двух программ пришли данные.

Номер порта – это число от 1 до 65 535. Для передачи такого числа по сети достаточно всего лишь двух байт. Рекомендуется использовать для своих целей порты с номерами более 1024, потому что среди меньших значений очень много зарегистрированных номеров, и у вашей программы увеличивается вероятность конфликта с другими сетевыми программами.

## Адрес узла

Если у вас только один компьютер, то в качестве адреса сервера клиент должен будет указать адрес 127.0.0.1. Этот адрес всегда показывает на ваш компьютер, что очень удобно при отладке сетевых приложений.

Если клиентская и серверная части приложения располагаются на разных компьютерах, то в клиентской части необходимо указать IP-адрес компьютера, на котором работает наша серверная часть. Выяснить его можно через утилиты командной строки winipcfg (для Windows 9\* и Me) и ipconfig (в Windows NT, 2000, XP). Вместо IP-адреса можно использовать и имя компьютера (в двойных кавычках). Например, имя "localhost" эквивалентно "127.0.0.1".

**Пример клиент-серверного приложения, который может быть использован для реализации сетевого чата**

Основой служат два компонента из стандартного пакета Delphi, это **ServerSocket** и **ClientSocket**. Они не всегда могут быть отображены в палитре Internet, и их нужно загрузить следующим образом:

выбрать меню: Component - Install Packages… - Add., далее нужно указать файл …\bin\dclsockets70.bpl.

## *Свойства компонента TServerSocket*

**Socket** - класс TServerWinSocket, через который Вы имеете доступ к открытым сокетным каналам. Далее мы рассмотрим это свойство более подробно, т.к. оно, собственно и есть одно из главных. Тип:TServerWinSocket;

**ServerType** - тип сервера. Может принимать одно из двух значений:stNonBlocking - синхронная работа с клиентскими сокетами. При таком типе сервера Вы можете работать с клиентами через события OnClientRead иOnClientWrite. stThreadBlocking - асинхронный тип. Для каждого клиентского сокетного канала создается отдельный процесс (Thread). Тип: TServerType;

**ThreadCacheSize** - количество клиентских процессов (Thread), которые будут кэшироваться сервером. Здесь необходимо подбирать среднее значение в зависимости от загруженности Вашего сервера. Кэширование происходит для того, чтобы не создавать каждый раз отдельный процесс и не убивать закрытый сокет, а оставить их для дальнейшего использования. Тип:Integer;

**Active** - показатель того, активен в данных момент сервер, или нет. Т.е., фактически, значение True указывает на то, что сервер работает и готов к приему клиентов, а False- сервер выключен. Чтобы запустить сервер, нужно просто присвоить этому свойству значение True. Тип: Boolean;

**Port** - номер порта для установления соединений с клиентами. Порт у сервера и у клиентов должны быть одинаковыми. Рекомендуются значения от 1025 до 65535, т.к. от 1 до 1024 - могут быть заняты системой. Тип: Integer;

**Service** - строка, определяющая службу (ftp, http, pop, и т.д.), порт которой будет использован. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам. Тип:string.

## *Методы компонента TServerSocket*

**Open** - Запускает сервер. По сути, эта команда идентична присвоению значенияTrue свойству Active;

**Close** - Останавливает сервер. По сути, эта команда идентична присвоению значенияFalse свойству Active.

## *События компонента TServerSocket*

**OnClientConnect**- возникает, когда клиент установил сокетное соединение и ждет ответа сервера (OnAccept);

**OnClientDisconnect**- возникает, когда клиент отсоединился от сокетного канала;

**OnClientError** - возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка;

**OnClientRead** - возникает, когда клиент передал берверу какие-либо данные. Доступ к этим данным можно получить через пеаедаваемый параметр Socket: TCustomWinSocket;

**OnClientWrite** - возникает, когда сервер может отправлять данные клиенту по сокету;

**OnGetSocket** - в обработчике этого события Вы можете отредактировать параметр ClientSocket;

**OnGetThread** - в обработчике этого события Вы можете определить уникальный процесс (Thread) для каждого отдельного клиентского канала, присвоив параметру SocketThread нужную подзадачу TServerClientThread;

**OnThreadStart, OnThreadEnd** - возникает, когда подзадача (процесс, Thread) запускается или останавливается, соответственно;

**OnAccept** - возникает, когда сервер принимает клиента или отказывает ему в соединении;

**OnListen** - возникает, когда сервер переходит в режим ожидания подсоединения клиентов.

## *Свойства компонента TClientSocket*

**Active –** показывает, открыт socket или нет. Тип: Boolean. Соответственно, True - открыт, а False - закрыт. Это свойство доступно для записи;

**Host –** строка (Тип: string), указывающая на хост-имя компьютера, к которому следует подключиться;

**Address –** строка (Тип: string), указывающая на IP-адрес компьютера, к которому следует подключиться. В отличие от **Host**, здесь может содержаться лишь IP. Отличие в том, что если Вы укажете в Host символьное имя компьютера, то IP адрес, соответствующий этому имени будет запрошен у DNS;

**Port** – номер порта (Тип: Integer (Word)), к которому следует подключиться. Допустимые значения - от 1 до 65535;

**Service –** строка (Тип: string), определяющая службу (ftp, http, pop, и т.д.), к порту которой произойдет подключение. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам;

**ClientType –** тип соединения. **ctNonBlocking** – асинхронная передача данных, т.е. посылать и принимать данные по сокету можно с помощью **OnRead** и **OnWrite**. **ctBlocking** - синхронная (одновременная) передача данных. События **OnRead** и **OnWrite** не работают. Этот тип соединения полезен для организации обмена данными с помощью потоков (т.е. работа с сокетом как с файлом).

## *Методы компонента TClientSocket*

Open открытие сокета (аналогично присвоению значения True свойству Active);

Close закрытие сокета (аналогично присвоению значения False свойству Active);

## *События компонента TClientSocket*

OnConnect как следует из названия, это событие возникает при установлении соединения. Т.е. в обработчике этого события уже можно начинать авторизацию или прием/передачу данных;

OnConnecting возникает при установлении соединения. Отличие от OnConnect в том, что соединение еще не установлено. Обычно такие промежуточные события используются для обновления статуса;

OnDisconnect возникает при закрытии сокета. Причем, закрытия как из Вашей программы, так и со стороны удаленного компьютера (либо из-за сбоя);

OnError Возникает при ошибке в работе сокета. Следует отметить, что это событие не поможет Вам отловить ошибку в момент открытия сокета (Open). Для того, чтобы избежать выдачи виндозного сообщения об ошибке, надо заключить операторы открытия сокета в блок try..except (обработка исключительных ситуаций);

OnLookup возникает при попытке получения от DNS IP-адреса указанного хоста;

OnRead возникает, когда удаленный компьютер послал Вам какие-либо данные. При возникновении этого события возможна обработка данных;

OnWrite возникает, когда Вам разрешена запись данных в сокет.

unit UnitMain;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ScktComp;

type

TForm1 = class(TForm)

ServerBtn: TButton;

ServerSocket: TServerSocket;

ClientSocket: TClientSocket;

PortEdit: TEdit;

HostEdit: TEdit;

NikEdit: TEdit;

TextEdit: TEdit;

ChatMemo: TMemo;

ClientBtn: TButton;

SendBtn: TButton;

procedure FormCreate(Sender: TObject);

procedure ServerBtnClick(Sender: TObject);

procedure ServerSocketClientConnect(Sender: TObject;

Socket: TCustomWinSocket);

procedure ServerSocketClientDisconnect(Sender: TObject;

Socket: TCustomWinSocket);

procedure ServerSocketClientRead(Sender: TObject;

Socket: TCustomWinSocket);

procedure SendBtnClick(Sender: TObject);

procedure ClientBtnClick(Sender: TObject);

procedure ClientSocketRead(Sender: TObject; Socket: TCustomWinSocket);

procedure ClientSocketConnect(Sender: TObject;

Socket: TCustomWinSocket);

procedure ClientSocketDisconnect(Sender: TObject;

Socket: TCustomWinSocket);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject);

begin

// предложенное значения порта

PortEdit.Text:='777';

// адрес при проверке программы на одном ПК ("сам на себя")

HostEdit.Text:='127.0.0.1';

// остальные поля просто очистим

NikEdit.Clear;

TextEdit.Clear;

ChatMemo.Lines.Clear;

end;

procedure TForm1.ServerBtnClick(Sender: TObject);

begin

If ServerBtn.Tag = 0 then

Begin

// клавишу ClientBtn и поля HostEdit, PortEdit заблокируем

ClientBtn.Enabled:=False;

HostEdit.Enabled:=False;

PortEdit.Enabled:=False;

// запишем указанный порт в ServerSocket

ServerSocket.Port:=StrToInt(PortEdit.Text);

// запускаем сервер

ServerSocket.Active:=True;

// добавим в ChatMemo сообщение с временем создания

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сервер создан');

// изменяем тэг

ServerBtn.Tag:=1;

// меняем надпись клавиши

ServerBtn.Caption:='Закрыть сервер';

end

else

Begin

// клавишу ClientBtn и поля HostEdit, PortEdit разблокируем

ClientBtn.Enabled:=True;

HostEdit.Enabled:=True;

PortEdit.Enabled:=True;

// закрываем сервер

ServerSocket.Active:=False;

// выводим сообщение в ChatMemo

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сервер закрыт.');

// возвращаем тэгу исходное значение

ServerBtn.Tag:=0;

// возвращаем исходную надпись клавиши

ServerBtn.Caption:='Создать сервер';

end;

end;

procedure TForm1.ServerSocketClientConnect(Sender: TObject; Socket: TCustomWinSocket);

begin

// добавим в ChatMemo сообщение с временем подключения клиента

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Подключился клиент.');

end;

procedure TForm1.ServerSocketClientDisconnect(Sender: TObject;

Socket: TCustomWinSocket);

begin

// добавим в ChatMemo сообщение с временем отключения клиента

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Клиент отключился.');

end;

procedure TForm1.ServerSocketClientRead(Sender: TObject;

Socket: TCustomWinSocket);

begin

// добавим в ChatMemo клиентское сообщение

ChatMemo.Lines.Add(Socket.ReceiveText());

end;

procedure TForm1.SendBtnClick(Sender: TObject);

begin

// проверка, в каком режиме находится программа

If ServerSocket.Active=True then

// отправляем сообщение с сервера (он под номером 0, поскольку один)

ServerSocket.Socket.Connections[0].SendText('['+TimeToStr(Time)+'] '+NikEdit.Text+': '+TextEdit.Text)

else

// отправляем сообщение с клиента

ClientSocket.Socket.SendText('['+TimeToStr(Time)+']'+NikEdit.Text+': '+TextEdit.Text);

// отобразим сообщение в ChatMemo

ChatMemo.Lines.Add('['+TimeToStr(Time)+']'+NikEdit.Text+': '+TextEdit.Text);

end;

procedure TForm1.ClientBtnClick(Sender: TObject);

begin

If ClientBtn.Tag=0 then

Begin

// клавишу ServerBtn и поля HostEdit, PortEdit заблокируем

ServerBtn.Enabled:=False;

HostEdit.Enabled:=False;

PortEdit.Enabled:=False;

// запишем указанный порт в ClientSocket

ClientSocket.Port:=StrToInt(PortEdit.Text);

// запишем хост и адрес (одно значение HostEdit в оба)

ClientSocket.Host:=HostEdit.Text;

ClientSocket.Address:=HostEdit.Text;

// запускаем клиента

ClientSocket.Active:=True;

// изменяем тэг

ClientBtn.Tag:=1;

// меняем надпись клавиши

ClientBtn.Caption:='Отключиться';

end

else

Begin

// клавишу ServerBtn и поля HostEdit, PortEdit разблокируем

ServerBtn.Enabled:=True;

HostEdit.Enabled:=True;

PortEdit.Enabled:=True;

// закрываем клиента

ClientSocket.Active:=False;

// выводим сообщение в ChatMemo

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сессия закрыта.');

// возвращаем тэгу исходное значение

ClientBtn.Tag:=0;

// возвращаем исходную надпись клавиши

ClientBtn.Caption := 'Подключиться';

end;

end;

procedure TForm1.ClientSocketRead(Sender: TObject;

Socket: TCustomWinSocket);

begin

// добавим в ChatMemo пришедшее сообщение

ChatMemo.Lines.Add(Socket.ReceiveText());

end;

procedure TForm1.ClientSocketConnect(Sender: TObject;

Socket: TCustomWinSocket);

begin

// добавим в ChatMemo сообщение о соединении с сервером

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Подключение к серверу.');

end;

procedure TForm1.ClientSocketDisconnect(Sender: TObject;

Socket: TCustomWinSocket);

begin

// добавим в ChatMemo сообщение о потере связи

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сервер не найден.');

end;

end.

Вывод: я ознакомился с понятием socket’ов и с помощью них реализовал сетевой чат в Delphi. Клиент должен знать IP-adress и порт на котором работает сервер, чтобы подключится к нему.