Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

*Лабораторная №5*

По Сети электронных вычислительных машин

Тема Создание сетевого приложения на основе TCP сокетов

Руководитель

Астафьев А.В.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИн-121

(группа)

Ермилов М.В.

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

**Лабораторная №7**

**Тема:** Создание сетевого приложения на основе TCP сокетов

**Задачи:**   
Передать серверу список строки и букву. Сервер должен вернуть

список строк исключая строки, начинающиеся на эту букву.

**Ход работы:**

Клиентская часть:

using System.Net.Sockets;

using System.Net;

using System.Threading;

using System.Text;

using System.Windows.Forms;

namespace work\_2

{

public partial class Form1 : Form

{

private static Socket Client; // Создаем объект сокета-сервера

private static IPHostEntry ipHost; // Класс для сведений об адресе веб-узла

private static IPAddress ipAddr; // Предоставляет IP-адрес

private static IPEndPoint ipEndPoint; // Локальная конечная точка

private static Thread socketThread; // Создаем поток для поддержки потока

private static Thread WaitingForMessage; // Создаем поток для приёма сообщений

private void startSocket()

{

// IP-адрес сервера, для подключения

string HostName = InputIP.Text;

// Порт подключения

string Port = InputPort.Text;

// Разрешает DNS-имя узла или IP-адрес в экземпляр IPHostEntry.

ipHost = Dns.Resolve(HostName);

// Получаем из списка адресов первый (адресов может быть много)

ipAddr = ipHost.AddressList[4];

// Создаем конечную локальную точку подключения на каком-то порту

ipEndPoint = new IPEndPoint(ipAddr, int.Parse(Port));

try

{

// Создаем сокет на текущей машине

Client = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

while (true)

{

// Пытаемся подключиться к удаленной точке

Client.Connect(ipEndPoint);

if (Client.Connected) // Если клиент подключился

{

// Позеленим кнопочку для красоты, чтобы пользователь знал, что соединение установлено

buttonConnect.Invoke(new Action(() => buttonConnect.Text = "Подключение установлено"));

buttonConnect.Invoke(new Action(() => buttonConnect.BackColor = Color.Green));

// Создаем новый поток, указываем на ф-цию получения сообщений в классе Worker

WaitingForMessage = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(GetMessages));

// Запускаем, в параметрах передаем листбокс (история чата)

WaitingForMessage.Start(new Object[] { listBox1 });

}

break;

}

}

catch (SocketException error)

{

// 10061 — порт подключения занят/закрыт

if (error.ErrorCode == 10061)

{

MessageBox.Show("Порт подключения закрыт!");

Application.Exit();

}

}

}

// Ф-ция, работающая в новом потоке: получение новых сообщенй ————

public static void GetMessages(Object obj)

{

// Получаем объект истории чата (лист бокс)

Object[] Temp = (Object[])obj;

ListBox ChatListBox = (ListBox)Temp[0];

// В бесконечном цикле получаем сообщения

while (true)

{

// Ставим паузу, чтобы на время освобождать порт для отправки сообщений

Thread.Sleep(50);

try

{

string Message = GetDataFromServer();

ChatListBox.Invoke(new Action(() => {

string[] Arr = Message.Split("\n");

if (Arr.Length > 0)

{

ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " Ответ:");

foreach (string str in Arr)

{

ChatListBox.Items.Add(str);

}

}

else

{

ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " В ответе нет строк!");

}

}));

}

catch { }

}

}

// Получение данных от сервера

public static string GetDataFromServer()

{

string GetInformation = "";

// Создаем пустое “хранилище” байтов, куда будем получать информацию

byte[] GetBytes = new byte[1024];

int BytesRec = Client.Receive(GetBytes);

// Переводим из массива битов в строку

GetInformation += Encoding.Unicode.GetString(GetBytes, 0, BytesRec);

return GetInformation;

}

public Form1()

{

InitializeComponent();

}

private void buttonConnect\_Click(object sender, EventArgs e)

{

socketThread = new Thread(new ThreadStart(startSocket));

socketThread.IsBackground = true;

socketThread.Start();

buttonConnect.Enabled = false;

buttonConnect.Text = "Ожидание подключения";

buttonConnect.BackColor = Color.Yellow;

}

private void buttonSend\_Click(object sender, EventArgs e)

{

SendDataToServer(InputText.Text);

// Добавляем в историю свое же сообщение + ник + время написания

listBox1.Items.Add(DateTime.Now.ToShortTimeString() + " Запрос отправлен!");

// Автопрокрутка истории чата

listBox1.TopIndex = listBox1.Items.Count - 1;

// Убираем текст из поля ввода

InputText.Text = "";

}

// Отправка информации на сервер

public static void SendDataToServer(string Data)

{

// Используем unicode, чтобы не было проблем с кодировкой, при приеме информации

byte[] SendMsg = Encoding.Unicode.GetBytes(Data);

Client.Send(SendMsg);

}

private void InputPort\_TextChanged(object sender, EventArgs e)

{

}

}

}

Серверная часть:

using System.Net.Sockets;

using System.Net;

using System.Threading;

using System.Text;

using System.Windows.Forms;

namespace work\_2

{

public partial class Form1 : Form

{

private static Socket Client; // Создаем объект сокета-сервера

private static IPHostEntry ipHost; // Класс для сведений об адресе веб-узла

private static IPAddress ipAddr; // Предоставляет IP-адрес

private static IPEndPoint ipEndPoint; // Локальная конечная точка

private static Thread socketThread; // Создаем поток для поддержки потока

private static Thread WaitingForMessage; // Создаем поток для приёма сообщений

private void startSocket()

{

// IP-адрес сервера, для подключения

string HostName = InputIP.Text;

// Порт подключения

string Port = InputPort.Text;

// Разрешает DNS-имя узла или IP-адрес в экземпляр IPHostEntry.

ipHost = Dns.Resolve(HostName);

// Получаем из списка адресов первый (адресов может быть много)

ipAddr = ipHost.AddressList[4];

// Создаем конечную локальную точку подключения на каком-то порту

ipEndPoint = new IPEndPoint(ipAddr, int.Parse(Port));

try

{

// Создаем сокет на текущей машине

Client = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

while (true)

{

// Пытаемся подключиться к удаленной точке

Client.Connect(ipEndPoint);

if (Client.Connected) // Если клиент подключился

{

// Позеленим кнопочку для красоты, чтобы пользователь знал, что соединение установлено

buttonConnect.Invoke(new Action(() => buttonConnect.Text = "Подключение установлено"));

buttonConnect.Invoke(new Action(() => buttonConnect.BackColor = Color.Green));

// Создаем новый поток, указываем на ф-цию получения сообщений в классе Worker

WaitingForMessage = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(GetMessages));

// Запускаем, в параметрах передаем листбокс (история чата)

WaitingForMessage.Start(new Object[] { listBox1 });

}

break;

}

}

catch (SocketException error)

{

// 10061 — порт подключения занят/закрыт

if (error.ErrorCode == 10061)

{

MessageBox.Show("Порт подключения закрыт!");

Application.Exit();

}

}

}

// Ф-ция, работающая в новом потоке: получение новых сообщенй ————

public static void GetMessages(Object obj)

{

// Получаем объект истории чата (лист бокс)

Object[] Temp = (Object[])obj;

ListBox ChatListBox = (ListBox)Temp[0];

// В бесконечном цикле получаем сообщения

while (true)

{

// Ставим паузу, чтобы на время освобождать порт для отправки сообщений

Thread.Sleep(50);

try

{

string Message = GetDataFromServer();

ChatListBox.Invoke(new Action(() => {

string[] Arr = Message.Split("\n");

if (Arr.Length > 0)

{

ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " Ответ:");

foreach (string str in Arr)

{

ChatListBox.Items.Add(str);

}

}

else

{

ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " В ответе нет строк!");

}

}));

}

catch { }

}

}

// Получение данных от сервера

public static string GetDataFromServer()

{

string GetInformation = "";

// Создаем пустое “хранилище” байтов, куда будем получать информацию

byte[] GetBytes = new byte[1024];

int BytesRec = Client.Receive(GetBytes);

// Переводим из массива битов в строку

GetInformation += Encoding.Unicode.GetString(GetBytes, 0, BytesRec);

return GetInformation;

}

public Form1()

{

InitializeComponent();

}

private void buttonConnect\_Click(object sender, EventArgs e)

{

socketThread = new Thread(new ThreadStart(startSocket));

socketThread.IsBackground = true;

socketThread.Start();

buttonConnect.Enabled = false;

buttonConnect.Text = "Ожидание подключения";

buttonConnect.BackColor = Color.Yellow;

}

private void buttonSend\_Click(object sender, EventArgs e)

{

SendDataToServer(InputText.Text);

// Добавляем в историю свое же сообщение + ник + время написания

listBox1.Items.Add(DateTime.Now.ToShortTimeString() + " Запрос отправлен!");

// Автопрокрутка истории чата

listBox1.TopIndex = listBox1.Items.Count - 1;

// Убираем текст из поля ввода

InputText.Text = "";

}

// Отправка информации на сервер

public static void SendDataToServer(string Data)

{

// Используем unicode, чтобы не было проблем с кодировкой, при приеме информации

byte[] SendMsg = Encoding.Unicode.GetBytes(Data);

Client.Send(SendMsg);

}

private void InputPort\_TextChanged(object sender, EventArgs e)

{

}

}

}

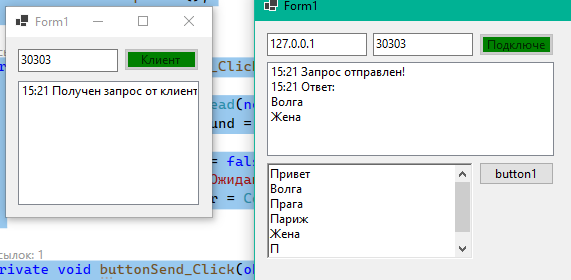


Рисунок 1 - пример работы программы