### MỤC LỤC

DANH SÁCH CÁC HÌNH VẾ DANH SÁCH CÁC BẢNG BIÊU

##### Chương1: MỚ ĐẦU

###### Lý do chọn đềtài:

Hiện nay, mạng Lan đã có những tiến bộ vượt bậc và ngày càng phố biến hơn trong đôi sống sinh hoat. Điều này làm cho nhu cầu liên lạc và trao đổi thông tin thông qua mạng Lan ngày càng lớn hơn. Chính vì vậy, chương trình Chat trên mạng Lan được xây dựng để đáp ứng phần nào những nhu cầu cấp thiết đó.

* 1. Mục đích của đề tài:

Xây dựng chương trình Chat hoạt động trong mạng Lan với các chức năng

cơ bản như: gởi tin nhăn, tạo một nhóm Chat và lưu thông tin bạn bè.

###### Đôi tượng và phạm vỉ nghiên cứu

* + 1. Đối tượng nghiên cứu

Tìm hiểu được cơ chế hoạt động của Socket và Thread trong .NET Framwork từ đó viết ứng dụng Chattrong mạng Lan.

* + 1. Phạm vỉ nghiên cứu

Chương trình Chat được xây dựng với khả năng gởi các được văn bản qualại giữa các user thông qua sự điều khiển của một Server trong mạng Lan.

# Chương 2: KIÊN THỨC ỨNG DỤNG

###### 2.1 Sơ lược về lập trình Socket:

2.1.1 Khái niệm Địa chỉ và công (Address & Port)

###### + Nguyên lý:

> Trong một máycó rất nhiều ứng dụng muốn trao đối với các ứng dụng khác thông qua mạng (ví dụ trên có 2 ứng dụng trong máy A muốn trao đổi với với 2 ứng dụng trên máy B).

> Mỗi máytính chỉ có duy nhất một đường truyền dữliệu (để gửi và nhận).

+ Vấnđề : Rất có thể xảy ra "nhằm lẫn" khi dữ liệu từ máy A gửi đến máy B thì không biết là dữ liệu đó gửi cho ứng dụng nào trên máy B?

###### % Giải quyết: Mỗi ứng dụng trên máy B sẽ được gán một số hiệu (mà ta

vẫn quen gọilà công : Port), số hiệu công nàytừ 1..65535. Khi ứng dụng trên máy A muốn gửi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu công (vào trường RemotePort) vào gói tin cần gửi. Trên máy B, các ứng dụng chỉ việc kiểm tra giá trị công trên mỗi gói tin xem có trùng với số hiệu công của mình (đã được gán - chính là giá trị Localport) hay không? Nếu bằng thì xử lý, còn trái lại thì không làm gì (vì không phải là của mình).

®& Như vậy: Khi cần trao đối dữ liệu cho nhau thì hai ứng dụng cần phải biết

thông tin tối thiểu là địa chỉ (Address) và số hiệu công (Port) của ứng dụng

kia.

* + 1. Lớp IPAddress

Trên Internet mỗi một trạm (có thể là máytính, máy In, thiết bị cà.) đều có một định danh duy nhất, định danh đó thường được gọi là một địa chỉ (Address). Địa chỉ trên Internet là một tập hợp gồm 4 con số có giá trị từ 0-255 và cách nhau bởi dấu chấm.

Đềthể hiện địa chỉ này, người ta có thể viết dưới các dạng sau:

% Tên : Ví dụ May01, Server, ....

+ Dịa chỉ IP nhưng đặt trong một xâu: "192.168.1.1", "127.0.0.1"

% Đặttrong một mảng 4 byte, mỗi byte chứa mộtsố từ 0-255. Ví dụ đề biểu diễn địa chỉ 192.168.1.1 với khai báo “byte[] DiaChi = new byte[4];”, ta có thể viết:

> DiaChi(0) = 192;

> DiaChi(1)= 168;

> DiaChi(2) = l1;

> DiaChi(3) = l1;

+ Hoặc cũng có thê là một số (long), có độ dài 4 byte. Ví dụ, với địa chỉ

192.168.1.1 ở trên thì giá trị đó sẽ là: 16885952 (đây là số ở hệ thập phân khi xếp liền 4 byte ở trên lại với nhau 00000001 00000001 10101000 11000000

I(Bye0) 1 168 192 (Byte 3)

+ Như vậy, để đổi một địa chỉ chuẩn ra dạng số ta chỉ việc tính toán cho

từng thành phần. Ví dụ: Đổi địa chỉ 192.168.1.2 ra số, ta tính như sau: 2\*256 ^3~+I\*256 ^2+ 168 \* 256 ^ I+ 192 \* 256 ^0

Trong .NET, IPAddress là một lớp dùng để mô tả địa chỉ này. Đây là lớp rất cơ bản được sử dụng khi chúng ta thao tác (truyền) vào các lớp như IPEndpoint, UDP, TCP, Socket...

##### Bảng 2-1: Các thành phần của lớp IpAddress

Thành viên Static Mô tả

Any Cung cấp một địa chỉ IP (thườnglà 0.0.0.0) để chỉ ra

rằng Server phải lắng nghe các hoạt động của Client trên tất cả các Card mạng (sử dụng khi xây dựng

Server). Thuộc tính này chỉ đọc.

Broadcasf Cung cấp một địa chỉ IP quảng bá (Broadcast, thường

##### là 255.255.255.255), ở dạngsố long.

Loopback Trả về một địa chỉ IP lặp (IP Loopback, ví dụ

127.0.0.1).

##### AddressFamily Trả về họ địa chỉ của địa chỉ IP hiện hành. Nếu địa chỉ

ở dạng IPv4 thì kết quả là Internetwork, và

InternetworkV6 nếulà địa chỉ IPv6.

Phương thức Mô tả

LP.Address(Inf604) Tạo địa chỉ IP từ một SỐ long.

IP.Address(Byfel]) Tạo địa chỉ IP từ một mảng Byte.

GŒef.AddressByfe () Chuyền địa chỉ thành mảng Byte.

HostToNeworkOrder(\_ Đảo thứ tự Byte của một số cho đúng với thứ tự Byte

trong địa chỉ IPAddress.

IsLoopbackQ Cho biết địa chỉ có phải là địa chỉ lặp hay không?

+ Ví dụ 1: Kiểm tra xem 192.168.1.300 có phải là địa chỉ IP hợp lệ không

private void KiemTra()

#### {

String IpI = "127.0.0.1”; String Ip2 = "999.0.0.1”;

MessageBox.Show(IPAddress. TryParse(IplI, new IPAddress(0))); MessageBox.Show (IPAddress. TryParse(Ip2, new IPAddress(1)));

#### ;

##### + Ví dụ 2: Chuyến địa chỉ hiện hành ra mảng byte và hiên thị từng thành

sphần trong mảng đó

private void KiemTra()

#### {

b));

IpAddress Ip3 = new IPAddress(16885952);

#### Bytel] b;

b = Ip3.GetAddressBytesQ);

MessageBox.Show( "Address: " & b(0) &"." & b(l1) &"."” & b(2) & "." &

* + 1. Lớp IPEndpoint

Trong mạng, để hai trạm có thể trao đôi thông tin được với nhau thì chúng

cần phải biết được địa chỉ (IP) của nhau và số hiệu công mà hai bên dùng đểtrao đổi thông tin. Lớp IPAddress mới chỉ cung cấp cho ta một về là địa chỉ IP (IPAddress), như vậy vẫn cònthiếu về thứ hai là số hiệu công (Port number). Như vậy, lớp IPEndpoint chínhlà lớp chứa đựng cả [PAddress và Port number.

Đối tượng IPEndpoint sẽ được dùng sau này để truyền trực tiếp cho các đối tượng UDP, TCP...

Bảng 2-2: Các thành viên của lớp IpEndPoint

Phương thức khởi tạo Mô tả

IPEndPoint(Tnf64, Inf32) Tạo một đôi tượng mới của lớp IPEndPoint,

tham số truyền vào là địa chỉ IP (ở dạng số) và công sẽ dùng đê giao tiếp.

IPEndPoint(LPAIddress, Inr32)

Tạo một đôi tượng mới của lớp IPEndPoint, Tham số truyền vào là một địa chỉ IPAddress và sô hiệu công dùng đê giao tiếp.

Thuộc tính Mô tả

##### Address Trả về hoặcthiết lập địa chỉ IP cho Endpoint (trả

về một đối tượng IPAddress).

AddressFamily Lây về loại giao thức mà Endpoint này đang sử

dụng.

Porí Lẫy hoặc gán số hiệu công của Endpoint.

Phương thức Mô tả

Creafe() Tạo một Endpoint từ một địa chỉ socket (socket

address).

ToString() Trả về địa chỉ IP và số hiệu công theo khuôn

dạng địa chỉ: công. Ví dụ: “192.168.1.1:8080”

2.1.4 Lớp UDP

Giao thức UDP (Ủser Datagram Protocol hay User Define Protocol) là một giao thức phi kết nối (connectionless) có nghĩa là một bên có thể gửi đữ liệu cho bên kia mà không cần biết là bên đó đã sẵn sàng hay chưa? (Nói cách khác là không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin). Giao thức này không tin cậy bằng giao thức TCP nhưng tốc độ lại nhanh và dễ cài đặt. Ngoài ra, với giao thức UDP ta còn có thể gửi các gói tin quảng bá (Broadcast) cho đồng thời nhiêu máy.

Trong .NET, lớp UDPClient (nằm trong namesapce System.Net.Sockets) đóng gói các chức năng của giao thức UDP.

Bảng 2-3: Các thành viên của lớp UDPCTient

Phương thức khởi tạo Mô tả

UdpClHenf () Tạo một đối tượng (thê hiện) mới của lớp

UDPClient.

UdpClHent (AddressFamily) Tạo một đôi tượng (thê hiện) mới của lớp

UDPClient Thuộc một dòng địa chỉ

(AddressFamnly) được chỉ định.

UdpCHemf (Inf32) Tạo một UdpClient và gắn (bind) một công cho

nó.

UdpCHenf (LPEndPoinf) Tạo một UdpClient và gắn (bind) một IPEndpoint

(gán địa chỉ IP và cổng) cho nó.

UdpClHenf(Inf32, Tạo một UdpClent và gán số hiệu công,

AddressFamily) AddressFamily

##### UdpClienf(String, Inf32) Tạo một UdpClient vàthiết lập với một trạm từ xa

mặc định.

Phương thức Môtả

BeginRcceive() Nhận dữ liệu Không đông bộ từ máy ở xa.

##### BeginSend0 Gửi không đồng bộ dữ liệu tới máy ở xa

##### Close() Đóng kết nối.

Connecf() Thiết lập một Default remote host.

##### EndReeeive0 Kết thúc nhận dữ liệu không đồng bộ ở trên

EndSend0 Kết thúc việc gửi dữ liệu không đồng bộ ở trên

Receive (reƒ IPEndPoinf) Nhận dữliệu (đồng bộ) do máy ở xa gửi. (Đồng bộ

có nghĩa là các lệnh ngay sau lệnh Recetve chỉ

được thực thi nếu Receive đã nhận được dữ liệu về

. Còn nếu nó chưa nhận được — dù chỉ một chút —

###### thì nó vẫn cứ chờ (blocking))

Send()

##### Gửi dữ liệu (đồng bộ) cho máy ở xa.

4+ Ví dụ 1: Tạo một UDPClient gắn vào công 10 và Gửi một gói tin "Hello" tới

một ứng dụng UDP khác đang chạy trên máy có địa chỉ là "127.0.0.1" và công 1000.

usingø SŠystem.Nct;

using System.Net.Sockets; public class Udp Test

{

const LOCAL PORT = 10; const REMOTE\_ PORT = 1000;

⁄ Tạo một UDP và gắn (Bind) vào cổng 10 UpdCleint Sender = new UdpClient(LOCAL \_PORT); privte void GửiDữ LiệuQ

#### {

/¡ Chuyển chuỗi "Hello there !" thành mảng byte để gửi đi

Byte[] msg = System. Text.Encoding.UTES.GetBytes("Hello there !");

⁄ Gửi vào cổng 1000 của máy 127.0.0.1

Sender.Send(msg, msg.Length, "127.0.0.1", REMOTE\_ PORT);

#### ;

—~

+ Ví dụ 2: Tạo một UDPClient gắn vào công 1000 và nhận đữ liệu từ ứng

##### dụng khác gửi đến.

usingø SŠystem.Nct; usinø System. Text;

using System.Net.Sockets; namespace Test

#### {

public class Udp Test

const LOCAL \_PORT = 1000;

const REMOTE\_PORT = 10;

UpdCTlient Receriver = new UdpClientLOCAL PORT); private void Nhận Dữ\_ Liệu(Q)

### {

IPEndPoint ep= new

IPEndPoint(IPAddress.Parse(”127.0.0.1”), 100); Byte[] msg = Receiver.Receive(ep);

String str;

str = Encoding.UTFS.GetString(msg); Chuyển byte -> String

MessageBox.ShowQGstr);

2.1.5. Lớp TCP (TCPClIient)

Mục đích của lớp UDPClient ở trên là dùng cho lập trình với giao thức UDP, với giao thức nàythì hai bên không cần phải thiết lập kết nối trước khi gửi do vậy mức độ tin cậy không cao. Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta còn dùng một giao thức khác, gọi là giao thức có kết nối : TCP (Transport Control Protocol). Trên Internet chủ yếu là dùng loại giao thức này, ví dụ như

Telnet, HTTP, SMTP, POP3... Để lập trình theo giao thức TCP, MS.NET cung cấp

hai lớp có tên là TCPClient và TCPLIstener. Bảng 2-4: Các thành phần của lớp TcpClient

Phương thức khởi tạo Mô tả

TcpChlienf() Tạo một đối tượng TcpClient. Chưa đặt thông SỐ gì.

TcpClHent(TPEndlPoinf)

Tạo một TepClient và gắn cho nó một EndPoint cục bộ. (Gán địa chỉ máy cục bộ và sô hiệu công đê sử dụng trao đôi thông tin về sau)

TcpChHienf(String,Inf32) Tạo một đôi tượng TepClient và kêt nôi đên một máy

có địa chỉ và sô hiệu công được truyên vào.

RemoteHost có thể là địa chỉ IP chuẩn hoặc tên máy.

Các thuộc tính Mô tả

Available Cho biết số byte đã nhận về từ mạng và có săn để đọc.

Chienf Trả về Socket ứng với TCPClient hiện hành.

Connected Trạng thái cho biết đã kết nối được đến Server hay

chưa?

Các hàm thành phần Mô tả

Close( Giải phóng đôi tượng TepClient nhưng không đóng kết

nôi.

Connecf(RemoteHost,

RemofePor£)

Kết nôi đên một máy TCP khác có Tên và sô hiệu công.

GefStream()

Trả về NetworkStream đề từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo StreamReader và StreamWriter để gửi và nhận dữ liệu dưới dạng xâu ký tự).

Khi đã gắn vào StreamReader và StreamWWriter rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readline, writeline tương ứng của các lớp này.

+ Từ các thành viên của lớp TepClient ở trên ta thây răng, việc kết nối và thực hiện gửi nhận rât đơn giản. Theo các trình tự sau:

> Bước l: Tạo một đối tượng TcpClient.

> Bước 2: Kết nối đến máy chủ (Server) dùng phương thức Connect.

> Bước 3: Tạo 2 đối tượng StreamReader (Receive)và StreamWriter (Send) và "nôi" với GetStream của cpPClient.

> Bước 4:

e© Dùng đối tượng StreamWriter.Writeline/Write vừa tạo ở trên để gửi

###### dữ liệu đi.

e© Dùng đối tượng StreamReader.Readline/Read vừa tạo ở trên để đọc

### dữ liệu về.

> Bước 5: Đóng kếtnối.

+ Nếu muốn gửi/nhận dữ liệu ở mức byte (nhị phân) thì dùng NetworkStream. (truyền GetStream cho NetworkStream).

2.1.6 Lớp TcpListener

TCPListerner là một lớp cho phép người lập trình có thể xây dựng các ứng dụng Server (Ví dụ như SMTP Server, FTP Server, DNS Server, POP3 Server hay server tự định nghĩa ....). Ứng dụng server khác với ứng dụng Client ở chỗ nó luôn luôn thực hiện lắng nghe và chấp nhận các kết nối đến từ Client.

Bảng 2-5: Các thành phần của lớp TepListener

Phương thức khởi tạo Mô tả

TcpListener ( Inf32) Tạo một TepListener và lăng nghe tại công chỉ

định.

TcpListener (TPEndPoint) Tạo một TepListener với giá trị Endpoint truyền

vào.

TcpListener(IPAddress,Inf32) Tạo một TepListener và lắng nghe các kết nổi

đến tại địa chỉ IP và công chỉ định.

Phương thức Môtả

AccepfSockef( ) Chấp nhận một yêu cầu kết nôi đang chờ.

AccepfTcpClienf() Chấp nhận một yêu cầu kết nỗi đang chờ. (Ứng

dụng sẽ dừng tại lệnh này cho đến khi nào có một kết nối đến — “Blocking”).

Pending() Cho biếtliệu có kết nỗi nào đang chờ đợi không

Sfarf() Bắt đầu lắng nghe các yêu cầu kết nồi.

S/op0 Dừng việc nghe.

+ Ví dụ: Tạo một server trong đó, khi có một client kết nôi đến thì server chuyền xâu đó thành chữ HOA và gửitrả lại cho Client.

10

using System.Net.Sockets; usingø SŠystem.Nct;

usingø System.IO;

using System.Net.Sockets; using System. Threading; namespace TcpListener Test

### {

public class frmServer

#### {

TcpListener TCPServer = new TcpListener(21);

bool Thoat = false;

TcpCHent[] Clients = new TepClient[101];

int CurrClient = 0;

public void Xu LyKet\_NoiQ)

#### {

int LastClient = CurrClient - 1;

TcpClient Con = Clients(LastClient);

StreamReader Doc = new StreamReader(Con.GetStream()); StreamWriter Ghi = new StreamWriter(Con.GetStream()); string S = null;

while (Thoat == false)

#### {

Application.DoEventsQ;

if (Doc.EndOfStream == false)

#### {

S5 = Doc.ReadLine();

/J/I Xử lý tại đây S =S.ToUpperQ); Ghi.WriteLine(S);

1II

//'Gởi lại cho Client...

Ghi.FlushQ;

#### ;

;

;

public void Nghe \_Ket\_NoiQ)

#### {

while (Thoat == false) {

Clients(CurrClient) = TCPServer.AcceptTcpClientQ;

CurrClient += l1;

Thread Th = new Thread(Xu LyKet \_No)); Th.StartQ;

\ Ỹ

#### ;

private void frm€Œlose(obJect s, FormClosingEventArgs e)

#### {

Thoat = true;

#### ;

private void Form1\_ Load(obJect s, EventArgs e)

#### {

TCPServer.StartQ;

Nghe Ket Noi

#### ;

public frmServer()

#### {

Load += Forml\_ Load; EFormClosing += frmClose;

#### ;

12

###### Sơ lược về lập trình đa luồng:

2.2.1 Khái niệm Luồng (Thread)

Một luông (Thread) là một chuỗi liên tiếp những sự thực thi trong chương trình. Trong một chương trình C#, việc thực thi bắt đầu bằng phương thức main() và tiếp tục cho đến khi kết thúc hàm main(). Cấu trúc nàyrất hay cho những chương trình có một chuỗi xác định những nhiệm vụ liên tiếp. Nhưng thường thì một chương trình cần làm nhiều công việc hơn vào cùng một lúc. Ví dụ trong Internet Explorer khi ta đang tải một trang web thì ta nhắn nút back hay một link nào đó, để làm việc này Internet Explorer sẽ phải làm ít nhấtlà 3 việc:

4% Lấy dữ liệu được trả về từ Internet cùngvới các tập tin đi kèm.

###### 4 Thể hiện trang Web.

+ Xem người dùng có nhập để làm thứ gì khác không.

Để đơn giản vấn đề này ta giả sử Internet Explorer chỉ làm hai công việc:

###### s Trình bày trang Web.

% Xem người dùng có nhập gì không.

Để thực hành việc này ta sẽ viết một phương thức dùng để lấy và thê hiện trang Web. Giả sử rằng việc trình bày trang Web mấtnhiều thời gian (do phải thi hành các đoạn Javascript hay các hiệu ứng nào đó ...). Vì vậy sau một khoảng thời gian ngắn khoảng 1/12 giây, phương thức sẽ kiểm tra xem người dùng có nhập gì không. Nếucó thì nó sẽ đuơc xử lí, nếu không thì việc trình bày trang sẽ được tiếp tục. Và sau 1/12 giây việc kiểm tra sẽ được lặp lại. Tuy nhiên viết phương thức này thì rất phức tạp do đó ta sẽ dùng kiến trúc event trong Window nghĩa là khi việc nhập xảy ra hệ thống sẽ thông báo cho ứng dụng băng cách đưa ra một event. Ta sẽ cập nhật phương thức để cho phép dùng các event:

$ Ta sẽ viết một bộ xử lí event để đáp ứng đối với việc nhập của người dùng.

+ Ta sẽ viết một phương thức để lấy và trình bày dữ liệu. Phương thức này

được thựcthi khi ta không làm bắt cứ điều gì khác.

13

Ta hãy xem cách phương thức lấy và trình bày trang web làm việc: đầutiên

nó sẽ tự định thời gian. Trong khi nó đang chạy, máy tính không thê đáp ứng việc nhập của người dùng .Do đó nó phải chú ý đến việc định thời gian để gọi phương thức kiểm tra việc nhập của người dùng, nghĩa là phương thức vừa chạy vừa quan sát thời gian. Bên cạnh đó nó còn phải quan tâm đến việc lưu trữ trạngthái trước khi nó gọi phương thức khác để sau khi phương thức khác thực hiện xong nó sẽ trả về

đúng chỗ nó đã dừng. Vào thời Window 3.1 đây thực sự là những gì phải làm để xử lí tình huống này. Tuy nhiên ở NT3.1 và sau đó là Windows 95 trở đi đã có việc xử lí đa luồng điều này làm việc giải quyết vấn đề tiện lợi hơn. Dưới đây chúngta sẽ tìm hiểu một vài lớp cơ bản trong ngôn ngữ lập trình C# và vấn đề đồng bộ hóa

(Synchronization) tronglập trình đa luồng.

* + 1. Khảo sát namespace System. Threading

Namespace System.Threading cung cấp một số kiểu dữ liệu cho phép bạn thực hiện lập trình đa luồng. Ngoài việc cung cấp những kiểu dữ liệu tượng trưng cho một luồng cụ thể nào đó, namespace này còn định nghĩa những lớp có thể quản lý một collection các luồng (ThreadPool), một lớp Timer đơn giản (không dựa vào GUI])và các lớp cung cấp truy cập được đồng bộ vào dữ liệu được chia sẽ sử dụng.

Bảng 2-6: Một số lớp của namespace System.Threading

Các lớp thành viên Mô tả

ITmterlocked Lớp này dùng cung cấp truy cập đồng bộ hóa vào dữ liệu được chia sẽ sử dụng (shared data).

Monifer Lớp này cung câp việc đông bộ hóa các đôi tượng luông

sử dụng khóa chốt (lock) và tín hiệu chờ (wait signal).

IMutex Lớp này cung cấp việc đồng bộ hóa sơ đăng có thể

được dùng đôi với Inter process synchron1zation.

Thread Lớp này tượng trưng cho một luồng được thi hành trong lòng Common Language Runtime. Sử dụng lớp này bạn có khả năng bổ sung những luồng khác trong cùng AppDomanm.

### 14

ThreadPool Lớp này quản lý những luông có liên hệ với nhau trong

cùng một Process nào đó.

Timer Cho biết một delegate có thể được triệu gọi vào một lúc được khai báo nào đó. Tác vụ wait được thi hành bởi

luồng trong thread pool.

WaitHandle Lớp này tượng trưng cho tất cả các đối tượng đồng bộ

hóa (cho phép multiple waIt) vào lúc chạy.

ThreqadSfart Lớp này là một delegate chỉ về hàm hành sự nào đó

##### phải đượcthi hành đầu tiên khi một luồngbắt đầu.

TimerCallBack Delegate đối với Timer.

JVaifCallBack Lớp này là một delegate định nghĩa hàm hành sự kêu

gọi lại (callback) đối với ThreadPool user work item.

* + - 1. Lớp Thread

Lớp sơ đăng nhất trong tất cả các lớp thuộc Namespace System.Threading là lớp Thread. Lớp này tượng trưng cho một vỏ bọc hướng đối tượng bao quanh một lộ trình thi hành trong lòng một AppDomain nào đó. Lớp này định nghĩa một số hàm thực thi (cả static lẫn shared) cho phép bạn tạo mới những luông từ luồng hiện hành, cũng như cho Sleep, Stop hay KiII một luồng nào đó.

Bảng 2-7: Các thành phần static của lớp Thread

Các thành phần Static Mô tả

CwrrenfThiread Thuộc tính read-only này trả về một quy chiếu về

luồng hiện đang chạy.

GŒefDufa() Đi lây vị trí từ slot được khai báo trên luồng hiện

hành đối với domain hiện hànhtrong luồng.

SetDafa() Cho đặt đểtrị lên slot được khai báo trên luông hiện

hành đối với domain hiện hành trong luồng

ŒefDormain() Đi lấy một qui chiếu về AppDomain hiện hành

GefDomainID() (hoặc mã nhận diện ID của domain này) mà luồng

hiện đang chạytrên đó.

15

Sleep0 Cho ngưng luồng hiện hành trong một thời gian nhất

Ngoài ra lớp Thread cũng hô trợ các thành viên cấp đôi tượng.

Bảng 2-8: Các thành viên cấp đối tượng của lớp Thread

Các lớp thành viên Mô tả

IsAlive Thuộc tính này trả về một trị boolean cho biết liệu xem

luồng đã khởi đông hay chưa.

IsBackground Đi lây hoặc đặt để giá trị cho biết liệu xem luồng là một

luồng nền hay không.

Name Thuộc tính này cho phép bạn thiết lập một tên văn bản

mang tính thân thiện đốivới luồng.

##### Priorify Đi lấy hoặc đặt để ưu tiên của một luồng. Có thể được

gán một trị lấy từ enumeration ThreadPriority (chăng hạn NormaL Lowest, Hiphest, BelowNormai, AboveNormai).

##### ThreadSfate Đi lẫy hoặc đặt để tình trạng của luồng. Có thế được

gán từ cnumeraton ThreadState (chăng hạn

Unstarted, Running, Wai(SleepJoin, Suspended, SuspendRequested, AbortRequested, Stopped).

Inferrup(0 Cho ngưng chạy luồng hiện hành.

Join() Yêu cầu luồng chờ đôivới luồng bị ngưng chạy.

Resune() Tiếp tục lại đối với một luông bị ngưng chạy.

##### Sfarf( Cho bắt đầu thi hành luồng được khai báo bởi delegate

ThreadStart.

Suspend() Cho ngưng chạy một luồng. Nêu luồng đã bị ngưng tồi,

một triệu gọi hàm Suspend0) sẽ không có tác dụng.

16

### Thao tác với luồng

Luồng được thao tác băng cách dùng lớp Thread năm trong Namespace System.Threading. Một thể hiện của luồng đại diện cho một luồng. Ta có thể tạo các luồng khác bằng cách khởi tạo một đối tượng Thread.

Giả sử rằng ta đang viết 1 trình biên tập hình ảnh đồ hoạ, và người dùng yêu cầu thay đôi độ sâu của màu trong ảnh. Ta bắt đầu khởi tạo một đối tượng luồng như sau:

/! entryPoint được khai báo trướclà I delegate kiêu ThreadStart

Thread depthChangeThread = new Thread(entryPoinnt);

Đoạn mã trên biểu diễn một hàm khởi tạo của Thread với một thông số chỉ

định điểm nhập của một luông. Đó là phương thức nơi luồng bắt đầu thi hành. Trong tình huống này ta dùng thông số là delegate, môt delegate đã được định nghĩa trong System.Threading gọi là ThreadStart, chữ kí của nó như sau:

public delegate void ThreadStartQ;

Thông số ta truyền cho hàm dựng phảilà I delegate kiểu này. Ta bắt đầu luồng bằng cách gọi phương thức Thread.Start() , giả sử răng ta có phương thức ChangeColorDepthQ:

void ChangeColorDepthQ

#### {

##### /¡ xử lí để thay đôi màu

;

Sắp xếp lại ta có đoạn mãsau :

ThreadStart entryPoint = new ThreadStart(ChangeColorDepth); Thread depthChangeThread = new Thread(entryPoint); depthChangeThread.Name = “Depth Change Thread”; depthChangeThread.StartQ;

Sau điểm này, cả hai luông sẽ chạy đồng bộ với nhau.

Trong đoạn mã này ta đăng kí tên cho luồng bằng cách dùng thuộc tính Thread.Name. Không cần thiết làm điều này nhưng nó có thê hữuích.

17

Lưu ý rằng bởi vì điểm đột nhập vào luồng (trong ví dụ này là ChangeColorDepthQ) ) không thê lấy bất kì thông số nào. Ta sẽ phải tìm một cách nào đó để truyền thông số cho phương thức nếu cần. Cách tốt nhất là dùng các trường thành viên của lớp mà phương thức này là thành viên. Cũng vậy phương

##### thức không thể trả về bất cứ thứ gì.

Mỗi lần ta bắt đầu một luồng khác, ta cũng có thê đình chỉ, hồi phục hay bỏ qua nó. Đình chỉ nghĩa là cho luồng đó ngủ (sleep) - nghĩa là không chạy trong I khoảng thời gian. Sau đó nó thê được phục hôi, nghĩa là trả nó về thời diễm mà nó bị định chí. Nếu luồng được bỏ, nó dừng chạy. Window sẽ huỷ tất cả dữ liệu màliên

hệ đến luồng đó, để luồng không thể được bắt đầu lại. Tiếp tục ví dụ trên, ta giả sử vì lí do nào đó luông giao diện người dùng trình bày một hộp thoại cho người dùng cơ hội để đình chỉ tạm thời sự đổi tiến trình. Ta sẽ soạn mã đáp ứng trong luồng mam:

depthChangeThread.Suspend(Q);

Và nếu người dùng được yêu cầu cho tiến trình được phục hồi:

depthChangeThread.Resume();

Cuối cùng nếu người dùng muốn huỷluồng : depthChangeThread.AbortQ;

Phương thức SuspendQ) có thể không làm cho luồng bị định chỉ tức thời mà có thể là sau một vài lệnh, điều này là để luồng được đình chỉ an toàn. Đối với phương thức AbortQ nó làm việc băng cách tung ra biệt lệ ThreadAbortException. ThreadAbortException là một lớp biệt lệ đặc biệt mà không bao giờ được xử lí. Nếu

##### luồng đó thực thi mã bên trong khối try, bất kì khối finally sẽ được thựcthi trước khi luồng bị huỷ. Sau khi huỷ luồng ta có thể muốn đợi cho đến khi luồng thực sự bị huỷ trước khitiếp tục luồng khác ta có thể đợi bằng cách dùng phương thức join():

depthChangeThread.AbortQ; depthChangeThread.JoinQ;

##### Join() cũng có một số overload khác chỉ định thời gian đợi. Nếu hết thờigian

này việc thi hành sẽ được tiếp tục. NÊu một luồng chính muốn thi hành một vài

18

hành động trên nó, nó cần một tham chiếu đến đối tượng luồng mà đại diện cho luồng riêng. Nó có thể lấy một tham chiếu sử dụng thuộc tính static - CurrentThread- của lớp Thread:

Thread myOwn Thread = Thread.CurrentThread; Có hai cách khác nhau màta có thể thao tác lớp Thread:

4 Ta có thể khởi tạo Ì đối tượng luồng , mà sẽ đại diện cho luồng đang

chạy và các thành viên thê hiện của nó áp dụng đến luồng đang chạy 4k Ta có thể gọi 1 số phương thức static . những phương thức nàysẽ áp

dụng đến luồng mà ta thực sự đang gọi phương thức từ nó. một phương thức static mà ta muốn gọi là Sleep(., đơn giản đặt luồng đang chạy ngủ một khoảng thời gian, sau đó nó sẽ tiếp tục.

* + 1. Đồng bộ hóa (Synchronization) trong lập trình đa luồng:

##### Đồng bộ hóa

Đôi khim có thể bạn muốn điều khiển việc truy cập vào một nguồn lực, chăng hạn các thuộc tính hoặc các hàm của một đối tượng, làm thế nào chỉ một mạch trình được phép thay đổi hoặc sử dụng nguồn lực đó mà thôi. Việc đồng bộ hóa được thể hiện thông qua một cái khóa được thiết lập trên đối tượng, ngăn không cho luồng nào đó truy cập khi mạch trình đi trước chưa xong công việc.

Trong phần này, ta sẽ là quen với cơ chế đồng bộ hóa mà Common

Language Runtime cung cấp: lệnh loek. Nhưng trước tiên, ta cần mô phỏng một nguồn lực được chia sẽ sử dụng bằng cách sử dụng một biến số nguyên đơn giản: counter.

##### Đề bắtđầu, ta khai báo biến thành viên và khởi gán về zero:

1nt counter = 0;

Bài toán được đặt ra ở đây như sau: luồng thứ nhất sẽ đọc trị counter (0) rồi gán giá trị này cho biến trung gian (temp). Tiếp đó tăng trị của temp rồi Sleep một khoảng thời gian. Luồng thứ nhất xong việc thì gắn trị của temp trả về cho counter và cho hiển thị trị này. Trong khi nó làm công việc, thì luông thứ hai cũng thực hiện một công việc giống như vậy. Ta cho việc này lập này khoảng 1000 lần. Kết quả mà ta chờ đợi là hai luồng trên đếm lần lượt tăng biến counter lên 1 và in ra kết quả 1, 2,3, 4... tuy nhiên ta sẽ xét đoạn chương trình dưới đây và thấy rằng kết quả hoàn toàn khác với những gì mà chúng ta mong đợi.

Đoạn mã của chương trình như sau:

usinø SŠystem;

using System. Threading; namespace TestThread

### {

19

public class Tester

### {

private int counter = 0;

static void Main(string[] args)

#### {

Tester t= new Tester(); t.DoTestQ;

\ Console.ReadLineQ;

Ỹ

public void Do Test(Q)

#### {

Thread tIÏ = new Thread(new ThreadStart(Incrementer)); t1.IsBackground = truc;

t1.Name = "Thread One";

t1.StartQ;

Console.WriteLine("Start thread {0}”, t1.Name);

Thread t2 = new Thread(new ThreadStart(Incrementer)); t2.IsBackground = truc;

t2.Name = "Thread Two"; (2.StartQ;

Console.WriteLine("Start thread {0}", t2.Name); t1.JoinQ;

t2.JoinQ);

Console.WriteLine( "All my threads are done.");

### ;

public void Incrementer()

#### {

try

#### {

while (counter < 1000)

#### {

Int temp = coun(er; tcmpT++T; Thread.Sleep(1); counfer = temp;

Console.WriteLIne(“Thread {0}. Incrementer: {1}”, Thread.CurrentThread.Name, counter);

#### ;

;

catch (ThreadInterruptedException)

### {

20

Console.WrieLIne(Thread {0} Interrupted! Cleanng up..."„, Thread.CurrentThread.Name);

### /

finally

#### {

Console.WriteLIne("Thread {0} Existing.", Thread.CurrentThread.Name);

#### ;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ; | ; |  | | | |
| ỉ |  |
|  |  | Ihread | Thread | Tuo. | lInceenenter: |
|  |  | Thread Ihread | Thread Thrzad | Ủng. Tun. | Incrementer: Incrementer: |
|  |  | Ihread | ThP†ead | Tuo. | lÍncPementEr: |
|  |  | Thread | Thread | Ủng. | TIncrenementer: |
|  |  | Ihread | Thrzad | Tun. | Incrementer: |
|  |  | Thread | Thread | Tuag. | TncPementer: |
|  |  | Thread | Thread | Ủng. | lITÏncrementer: |
|  |  | Ihread | Thrzad | Tuo. | lIncenmenter: |
|  |  | Thread Ihread | Thread Thrzad | Ủng. Ủng. | IncrPementer: Incrementer: |
|  |  | Ihread | ThPead | Tuo. | lÍncPementeEr: |
|  |  |  | Thread  Thread | Tuao.  Ủng. | Tncrementer:  Incrementer: |
|  |  |  | II (0l 20-0) 5o  Thread Ủng. Incrementer: | | |
|  |  |  | Th#ead | Ủng. | Incrementer: |
|  |  | Thread Ihread | Thread Thrzad | Tuag. Ủng. | TIncrPementer: Incrementer: |
|  |  | Ihread | Thread | Ông. | lIncrenenter: |
|  |  | Theread | Thread | Tuao. | TIncrenementer: |
|  |  | Ihread | Thrzad | Ủng. | Incrementer: |
|  |  | Thread | ThP†Ead | Ủng | Exlstind. |
|  |  |  | II: | 6:0 | ( ai no |
|  |  |  | Thread | Tun | Exilstindg. |

lị I Ê TiiT

##### Hình 2-1: Kết quả chương trình không sử dụng đồng bộ hóa

Do đó ta cần phải đồng bộ hóa việc truy cập đối tượng counter.

##### C# cung cấp đối tượng Lock để thưc hiện công việc đồng bộ hóa này. Một

lock sẽ đánh dâu một critical section trên đoạn mã đồng thời cung cấp việc đồng bộ

hóa đối với đối tượng được chỉ định khi lock có hiệu lực. Cú pháp sử dụng một Lock yêu cầu khóa chặt một đối tượngrồi thi hành một câu lệnh hoặc một khối lệnh rồi sẽ mở khóa ở cuối câu hoặc khối lệnh đó. C# cung cấp hỗ trợ trực tiếp khóa chặt

thông qua từ chốt lock. Ta sẽ tra qua theo một đối tượng qui chiếu và theo sau từ

#### chốt là một khối lệnh

lock(express1on) statement-blockÄ

Trong ví dụ trên, để có được kết quả như mong muốn, ta sẽ sửa hàm Incrementer lại như sau:

21

try

lock (this)

#### {

while (counter < 1000)

### {

Int temp = counter; tempT+T; Thread.Sleep(1); counter = temp;

Console.WriteLine(TThread {0}. Incrementer: {1?",

Thread.CurrentThread.Name, counfter);

`

%

##### /¡ Các khối catch và fïnally không thay đồi

Kết quả thu được sẽlà:

B8 íi=:///C/U sers/Du Tue/Desktop/Dao An-Chat/Tr J= =

|  |  |  |
| --- | --- | --- |
| In | In In Tnœwementerz | 123 |
| Thread | li II) N0ì 22-10-0005 0 | 1143 |
| Th“ead | l1: 0(10)(0)- 0000032200-011. | Lực |
| HD». | Th#ead na. Incementer: | 32B |
| Threzad | Thread Ủng. TIncrementer: | 1”? |
| Thread | Th#ead Ủng. Incrementer: | 32B |
| Thrzad | Thread Ủne. Incrementer: | 1ụ |
| Threead | Thread ng. Incremente+: | kẻ) |
| L§U»:T- Tu | Thead Ủng. Incrementer: | Lá ! |
| Thread | Thread Ône. IncrPementer: | 182 |
| LỆj ¡| | li 0ï 0l: 2a. i12. | 83 |
| HD »-t- tui | II: N1: 0000032 0á0ì | tả: |
| Threzad | Thread Ủng. TIncrementer: | bá |
| Thread | Th#ead Ủng. Tncrementer: | Kác1 |
| ID». | l1) 000020000) 2 | L..t |
| Thread | li. II): N0¡ 2210-0105 0 | bis4s |
| Thread | Thead Ủng. Incrementer: | 309 |
| HD». | Thread Ủne. Incrementer: | Làn.) |
| Threead | Thread ng. TIncremente+: | Làn! |
| LỆ »2:t- ru | Th#ead Ủng. Incrementer: | tà" |
| Thread | Thread Ông. Incrementer: | 13 |
| Lãj ¡| | li 0-00 (202. | „41 |
| Thread | Thead Ủng. Incementer: | tập, |
| Thread | li II) N0ì 22-10-0005 0 | Là» |
| Lãi  HD». | Thread ng. TIncrementer+:  Th#ead na. Incementer: | LàNi 1E |
| Threzad HD »2:t- ru | Thread Ủng. TIncrementer: Th#ead Ủng. Incrementer: | thà:  IN 2.12 |
| Thrzad | Thread Ủne Existind. |  |
| Thread | Thread Tụo Existind. |  |

)

Nhị),

theads ame dũne.

IxÍ

Hình 2-2: Kết quả chương trình sử dụng đồng bộ hóa

22

Việc đồng bộ các luồng là quan trọng trong các ứng dụng đa luông. Tuy

### nhiên có mộtsố lỗitỉnh vi và khó kiểm soát có thể xuất hiện cụ thể là deadlock và

race condition.

* + - 1. Deadlock

Deadlock là một lỗi mà có thê xuất hiện khi hai luông cần truy nhập vào các tài nguyên bị khoá lẫn nhau. Giả sử một luồng đang chạy theo đoạn mãsau, trong đó A, B là hai đối tượng tham chiếu mà cả hai luồng cầntruy nhập :

###### lock (A)

{

// do something

###### lock (B)

{

// do something

;

`

Vào cùng lúc đó 1 luông khác đang chạy:

###### lock (B)

{

// do something

###### lock (A)

{

// do something

#### ;

;

Có thê xảy ra biến cô sau: luồng đầu tiên yêu cầu một lock trên A, trong khi vào cùng thời điểm đó luồng thứ hai yêu cầu lock trên B. Một khoảng thời gian ngắn sau, luồng A gặp câu lệnh loek(B), và ngay lập tức bước vào trạng thái ngủ, đợi cho lock trên B được giải phóng. Và tương tự sau đó, luồng thứ hai gặp câu lệnh

23

loek(A) và cũngrơi vào trạng thái ngủ chờ cho đến khi lock trên A được giải phóng

.Không may, lock trên Á sẽ không bao giờ được giải phóng bởivì luồng đầu tiên mà đã lock trên A đang ngủ và không thức dậy cho đến khi lock trên B được giải phóng điều này cũng không thể xảy ra cho đến khi nào luông thứ hai thức dậy. Kết quả là deadlock. Cả hai luồng đều không làm gì cá, đợi lẫn nhau để giải phóng lock. Loạilỗi này làm toàn ứng dụngbị treo, ta phải dùng Task Manager đề hủy nó.

Deadlock có thể được tránh nếu cả hai luồng yêu cầu lock trên đối tượng theo cùng thứ tự ..Trong ví dụ trên nếu luông thứ hai yêu cầu lock cùng thứ tự với luồng đầu, A đầutiên rồi tới b thì những luồng màlock trên a đầu sẽ hoàn thành nhiệm vụ của nó sau đó các luông khácsẽ bắt đâu.

* + - 1. Race condifion

Race condition là cái cái gì đó tỉnh vi hơn deadlock. Nó hiếm khi nào dừng việc thực thi của tiến trình , nhưng nó có thể dẫn đến việc dữ liệu bị lỗi. Nói chung nó xuất hiện khi vài luồng cố gắng truy nhập vào cùng một dữ liệu và không quan tâm đến các luồng khác làm gì để hiểu ta xem ví dụ sau:

Giả sử ta có một mảng các đối tượng, mỗi phần tử cần được xử lí bằng một cách nào đó, và ta có một số luồng giữa chúng làm tiến trình này. Ta có thể có một đối tuợng gọi là ArrayController chứa mảng đối tượng và một số int chỉ định số phần tử đượcxử lí .tacó phương thức:

1nt GetObJect(nt Index)

#### {

// trả về đối tượng với chỉ mục được cho

#### ;

Và thuộc tính read/write

int ObJectsProcessed

### {

// chỉ định bao nhiêu đối tượng đượcxử lí

### 24

sau:

Bây giờ mỗi luồng mà dùng đểxử lí các đối tượng có thê thi hành đoạn mã

lock(ArrayController)

#### {

1nt nextlndex = ArrayController.ObJectsProcessed; Console.WriteLine( ObJect to be processed next 1s ” + NextIndex);

++ArrayController.ObJectsProcessed;

obJect next = ArrayController.GetObJjectQ;

##### }

ProcessObJect(nexÐ);

Nếu ta muốn tài nguyên không bị giữ quálâu , ta có thê không giữ lock trên ArrayController trong khi ta đang trình bày thông điệp người dùng .Do đó ta viết lại đoạn mã trên:

lock(ArrayController)

##### {

1nt nextlndex = ArrayController.ObJectsProcessed;

##### }

Console.WriteLine(”ObJect to be processed next1s ” + nextlndex); lock(ArrayController)

##### {

++ArrayController.ObJectsProcessed;

obJect next = ArrayController.GetObJectQ;

##### ì

ProcessObJect(nexÐ);

Ta có thể gặp một vấn đề. Nếu một luồng lấy lẫy đối tượng (đối tượng thứ II trong mảng) và đi tới trình bày thông điệp nói về việc xử lí đối tượng này. Trong khi đó luồng thứ hai cũng bắt đầu thi hành cũng đoạn mã gọi ObjectProcessed, và

25

##### quyết định đối tượng xử lí kế tiếp là đối tượng thứ 11, bởi vì luồng đầu tiên vẫn

chưa được cập nhật.

ArrayController.ObjectsProcessed trong khi luồng thứ hai đang viết đến màn hình rằng bây giờ nó sẽ xử lí đối tượng thứ 11, luồng đầu tiên yêu cầu một lock khác trên ArrayController và bên trong lock này tăng ObJectsProcessed. Không may, nó quá trễ. Cả hai luồng đều đangxử lí cùng một đối tượng và loại tình huống nàyta gọi là Race Condition.

## Chương 3: PHÂẦN TÍCH THIẾT KẾ CHƯƠNG TRÌNH

###### Phân tích

* + 1. Phân tích nhu cầu thực tiễn:

Hiện nay, mạng Lan pháttriển rất mạnh cả trong trường học, các cơ quan tô chứcvà ở cả các hộ gia đình. Chính điều đó kéo theo nhu cầu liên lạc trao đổi thông tin trong mạng Lan cũng phát triển theo. Chính vì vậy, một chương trình Chat phục vụ cho nhu câu liên lạc, trao đôi thông tin trong mạng Lan là rất cần thiết.

* + 1. Yêu cầu đề ra:

Yêu cầu đặt ra là xây dựng chương trình Chat hoạt động trong mạng Lan sử dụng Socket và Multithreading đòi hỏi các chức năng nghiệp vụ sau:

3k Chat giữa hai người với nhau: Hai người gởi thông điệp qua lại cho nhau.

3È Chat giữa một nhóm người: Một người đứng ra tạo một nhóm Chat và mời các thành viên khác tham gia thảo luận.

% Một User có khả năng thêm và xóa một người vào Friend List của mình đề có thê liên lạc một cách dê dàng.

26

3.1.3. Mô hình dữliệu ở mức quan niệm:

Users

ID Spi> Integer

<M>

¡ Usemame Varable characters (20) <M>

\_=8,n Password Variable characters (15) <M> State Shorft integer

Fullname Varable characters (50) <M> Email Variable characters (50)

í 0,n Address Variable characters (S0)

FriendList ;

ID

<pi>

z 0,n

0 7 \* :

⁄ 0n GroupDetail

⁄# Relationship

OfflineMessage q 9í SendTime Date & Time <M> tr Grou Sucũ

Message Variable characters (500) <M> l l

GrouplÐ <pi> Integer <M> GroupName Variable characters (30) <M> CreateTime Date & Time <M>

GroupID <pi>

GroupID <pi>

Hình 3-1: Mô hình dữliệu ở mức quan niệm

* + 1. Phân tích các thành phần xữlý:
       1. Các xửlý liên quan đến đăng nhập

+ Xử lý đăng nhập:

> Mô hình xử lý:

ZiẾ

## 

Đăng nhập thành công

Z8

k

k7 đ

+ Màn hình đăng nhập |Màn hình chính Serwer | Table Usrs EriendList Table OfflineMessage T†

Người dùng Fiend

Nhập Usamamae và Password

\_——

Yêu cầu đăng nhập

Gời Usemame và Password C Kim tra Username và P4ssword

Kim tra Username vàHH

\_Username và Pasword hợplệ L

Kiểm tra trạng thái đăng nhập. L| Kiểm tra trạng thái đăng nhập

User chưa đăng nhập F}m Cập nhật trạng thái đăng nhập

Danh sách các Friend

Danh sách các:Friend L

II các Friend đang Online và Offline

Tập hợp OfflineMessage của người dùng

Tập hợp các Offline Message LỊ

Thông báo đăng nhập t|

Gời danh sách Friend và OiflineMe sage

nh công Thông báo cho các Friend đang Online của người dùng này

Hiễn thị màn hình chính

VỊ

Hình 3-2: Mô hình xử lý đăng nhập

> Mô tả:

Khi một User yêu cầu đăng nhập hệ thống, Client sẽ gởi Username và Password cho Server. Server sẽ kiểm tra Username và Password có hợp lệ hay không. Sau đó, Server sẽ kiểm tra Username này đã đăng nhập chưa. Nếu đăng nhập thành công, Server sẽ lẫy danh sách các Friend đang Offline và Online của Ủser kèm theo danh sách các tin nhắn Offline (nếu có) và gởi cho User và cập nhậtlại trạng thái đăng nhập của User. Đồng thời cũng gởi thông báo đến các sers khác có Friend là Ủser này mới Online.

+ Xử lý đăng xuất:

> Mô hình xửlý:

28

Người dùng đăng xuất khỏi chương trình

/ G

k# tờ

¬ Màn hình chính Màn hình đăng nhập. Senwer | Table EriendLis Table GroupDetail Table Users

VN ——

—=

\_ ——...

Ị —\_"` —

Người dùng Users

Chọn chức năng đăng xuất -

† Thông báo cho Server là người dùng đăng xuất

Lấy danh sách các Usrs có Friend là User này

"I1 Danh sách các Users

Hiễn thị màn hình đăng nhập Lấy danh sách các Users có Friend là Users này đang Online

Lấy danh sách các Usrs có trong các Group mà User này tham gia

Thông báo cho các Usrs có liên quan là Usr này đã đăng xuất

Gập nhật lại các Group mà Usar tham gia

1

Gập nhật lại trạng thái đăng nhập ]

##### Hình 3-3: Mô hình xử lý đăng xuất

> Mô tả:

Khi một Ủser yêu cầu đăng xuất lại hệ thống, Client sẽ hiển thị lại màn hình đăng nhập và đồng thời gởi thông báo đang xuất đến Server. Server sẽ gởi thông báo đăng xuất tới các Users có Friend là User này. Bên cạnh đó, Server kiểm tra tất cả các Groups mà User này đang tham gia. Server sẽ gởi thông báo hủy đến các Group mà User này là người khởi tạo và sẽ gởi thông báo đăng xuất đến các Group mà User này chỉ tham gia với tư cách là thành viên. Cuôi cùng, Server sẽ cập nhật lại trạng thái đăng nhập của ser.

##### Các xử lý đối với việc gởi tin nhắn

###### + Gởi tin nhắn Online:

> Mô hình xử lý:

^Ó

Gởi tin nhắn Online

fSài 7=... ƠỰỢ

xe 4

=s Màn hình chính Server Table Users t

Ngưởi gởi Người nhận

Nhập tỉn nhắn

Yêu cầu gởi tin nhắn

®T| Hiển thị tin nhắn

Gởi tin nhắn vả tên người nhận

Kiểm tra người nhận Online

Ngưởi nhận Online IƑ

Kiểm tra Online

Gời tin nhắn và tên người gởi

Š =

##### Hình 3-4: Mô hình xử lý gởi tin nhắn Online

> Mô tả:

Khi người dùng nhập tin nhắn và nhắn phím Enter (hoặc button Senđ), Client sẽ hiển thị tin nhắn trên Tab Chat đồng thời gởitin nhắn và tên người nhận lên cho Server. Server sẽ kiểm tra xem người nhận có phải đang Online hay không. Nếu người nhận đang Online, Server sẽ gởi tin nhắn và tên người gởi. Khi nhận được tin nhắn, Client sẽ hiển thị tin nhắn thông qua một Tab Chat (Tab này sẽ được mở nếu nó chưa có trước đó).

##### + Gởi tin nhắn Offline:

> Mô hình xử lý:

30

Gởi tin nhắn Offline

k7

##### JN

Ngưởi gởi

[

Màn hình chính

Server Table Users

Table OfflineMessage

Nhập tin nhắn

Yêu cầu gởi tin nhắn Hiển thị tin nhắn

Gởi tin nhắn và tên người nhận -

Kiểmtra người nhận Online Kiểm tra Online

¡1

IƑ

Người nhận không Online

Lưu tin nhắn |

##### Hình 3-5: Mô hình xử lý gởi tin nhắn Offline

> Mô tả:

Khi người dùng nhập tin nhắn và nhắn phím Enter (hoặc button Senđ), Client sẽ hiển thị tin nhắn trên Tab Chat đồng thời gởitin nhắn và tên người nhận lên cho Server. Server sẽ kiểm tra xem người nhận có phải đang Online hay không. Nếu người nhận đang Offline, Server sẽ lưu tin nhắn cùng tên người gởi vào bảng OfflineMessage để gởi cho người nhận ở lần đăng nhập tiếp theo.

##### Các xử lý đối với FriendList

###### + Thêm một Friend vào FriendList:

> Mô hình xửlý:

3l

Thêm Friend vào FriendList

Màn hình chính

Màn hình thêm Eriend

Server

Table Users Table EriendList

Người dùng ị T \_

Yêu cầu thêm Friend

Km thị màn hình thêm Friend

Yêu cầu nhập tến Fiend cần thêm

Nhập tên Friend

Yêu cầu thêm Friend

Gởi tên Friend

Kiểm tra tồn tại Kiểm tra tôn tại

Friend tồn tại T

Kiểm tra Friend tồn tại trong FriendList

Chưa tổn tại

Thêm Friend vào FriendList

1Ì Kiểm tra tồn tại

IƑ

Thông báo thêm Friend thành công

Cập nhật màn hình chính

Thêm Friend thành công

Hình 3-6: Mô hình xử lý thêm Friend

> Mô tả:

Khi người dùng chọn chức năng thêm một Friend vào FriendLIist, màn hình thêm FriendLIst sẽ được mở ra. Người dùng sẽ nhập tên của Friend và sau đó ClHient sẽ gởi tên Friend này lên cho Server. Trước tiên, Server sẽ kiểm tra Friend này có tồn tại hay không. Tiếp theo, sẽ kiểm tra Friend này đã được thêm vào FriendList trước đó hay chưa. Nếu Friend này chưa có trong FriendList, Server sẽ thêm Friend này vào FriendList của người dùng. Cuối cùng, Server sẽ gởi kết quả của công việc về cho Client. Dựa vào kết quả nhận được, Client sẽ thông báo cho người dùng biết là việc thêm thành công haythất bại (có hai nguyên nhân thất bại là Friend không tồn tại và Friend đã có trong FriendList rồi).

###### + Xóa một Friend ra khỏi FriendList:

> Mô hình xử lý:

32

Xóa Friend ra khỏi FriendList a

Người dùng

Màn hình chính Màn hình xóa Eriend Server Table Users Table EriendList

###### 

Yêu cầu xóa Friend

Hiển thị màn hình xóa Friend

Yêu cầu nhập tên Friend cần xóa

Nhập tên Friend

ba Yêu cầu xöa' Friend ra khỏi FriendList

Tên Friend

Kiểm tra tồn tại

UŨ Kiểm trạ tồn tại

Friend tồntại

Kiểmtra Friend tồntại trong FriendList

Kiểm tra tồntại

Eriend tồn tại trong FriendList l

Xöa Friend ra khỏi FriendList

\_\_ Xóa thành công

Thông báo xóa thành công

Cập nhật màn hình chính

Hình 3-7: Mô hình xử lý xóa Friend

> Mô tả:

Khi người dùng chọn chức năng xóa một Frlend vào FriendLIist, màn hình thêm FriendLIst sẽ được mở ra. Người dùng sẽ nhập tên của Friend và sau đó ChHient sẽ gởi tên Friend này lên cho Server. Trước tiên, Server sẽ kiểm tra Friend này có tồn tại hay không. Tiếp theo, sẽ kiểm tra Friend này đã được thêm vào FriendList trước đó hay chưa. Nếu Friend này đã có trong FriendList, Server sẽ xóa Friend này ra khỏi FriendList của người dùng. Cuối cùng, Server sẽ gởi kết quả của công việc về cho Client. Dựa vào kết quả nhận

##### được, Client sẽ thông báo cho người dùng biết là việc Xóa thành công

hay thất bại (có hai nguyên nhânthất bại là Friend không tôn tại và Friend chưa có trong FriendList) đồng thời cập nhật lại màn hình chính nếu cần.

> Ghi chú:

Chức năng này có thể được gọi khi người dùng nhấn phím Delete trong Listbox FriendLIst.

* + - 1. Các xử lý đối với Group Chat

###### % Khi User tạo Group:

> Mô hình xửlý:

33

Tạo Group.

## ÁY

Màn hình chính.

Màn hình tạo group. Server

[TablĐe LUAsers

l®.)

## ~5

Người tạo group. Người được mời

Yêu cầu tạo Group

TIỊ Hiễn thị màn hình tạo Group

Yêu cầu chọn các User được mời vào Group.

Nhập tên các User cần mời vào Group.

'Yêu cầu tạo Group Mở Tab Group Chat

Gởi tên các Users và tên Group

ˆ Kiểm tra tồn tại

Tập hợp các Ur tồn tại |

Gởi thông báo mời cho các User

Hình 3-8: Mô hình xử lý tạo Group

> Mô tả:

Khi người dùng chọn chức năng Invite Group thì màn hình tạo Group sẽ xuất hiện. Người dùng nhập danh sách các Users muốn mời và có thể nhập thêm Invite Message rồi nhắn Invite. Client sẽ tự phát sinh ra tên Group và sẽ gởi kèm nó với danh sách các Users muốn mời đồng thời cũng mở thêm một Tab Group Chat. Server sẽ lọc ra các sers đang Online trong danh sách các sers được mời. Sau đó, Server sẽ gởi lời mời kèm theo tên người mời đến các Users được TỜI.

###### + Khi User đồng ý gia nhập Group:

> Mô hình xửlý:

34

Đồngý gia nhập Group

Ọ Ọ

Màn hình chính | Server Table GroupChat Table GroupDetail +

Người dùng Userđã tham gia Group

Đồng ý gia nhập Group. Mờ Tab GroupChat

Đồng ý tham gia Group. Kiểm tra Group

Group chưa tồn tại Thêm Group

Các Usr đã tham gia Group

CậpC hợpGP CỐcácC Usr@GI thamEm B8gia. GìT0 Tập hợp các Userđã thamgia Group Ụ

Gâgiiäf'äB'Gidig'Giát Thêm người dùng này vào Group Chat

Thông báo cho các Usr đã tham gia trước cập nhật lại Tab Group Chat

Hình 3-9: Mô hình xử lý đồng ý gia nhập Group

> Mô tả:

Khi người dùng đồng ý lời mời gia nhập nhóm, Client sẽ gởi thông báo đồng ý cho Server. Server sẽ kiểm tra xem Group này đã được lưu hay chưa (Group chưa được lưu nếu chưa có người dùng nào đồng ý tham gia). Sau đó, Server sẽ lưu người dùng này vào GroupDetail. Cuối cùng, Server sẽ gởi tên tất cả các Users đã tham gia Group cho người dùng, đồng thời cũng gởi thông báo đến cho các Users trong Group là người dùng này đã đồng ý gia nhập nhóm.

###### $ Khi User thoát khỏi Group:

> Mô hình xửlý:

35

Rời khỏi Group

@

=

s, ='

NÀY Màn hình chính Server Table GroupDetail :

Người dùng .nẽ T=m=—= — UeR

Đóng Tab Group Chat

Cập nhật lại màn hình chính

IS thông báo thoát khỏi Group kèm theo tên Group

\_ Lầy danh sách các Users có trong Group Danh sách các User có trong Group LỊ

Thông báo đến các Usrstrong Group

Cập nhật lại Group Ú

Hình 3-10: Mô hình xử lý thoát khỏi Group

> Mô tả:

Khi một người dùng tắt Tab Group Chat, Client sẽ gởi thông báo thoát khỏi Group cho Server. Server trước tiên kiểm tra người dùng này có phải là người đã tạo Group hay không. Nếu không phải, Server sẽ gởi thông báo tới các Users khác trong Group. Cuối cùng, Server sẽ cập nhật lại GroupDetail.

###### + Khi User hủy Group:

> Mô hình xửlý:

36

Hủy Group

aä [ = =xY [ ='i đã

\

+ Màn hình chính

(\37

| Server | Table GroupChat Table GroupDetail =

ZN mi

ị —— =——— ⁄

Người dùng Usrs

Đóng Tab Group Chat. Cập nhật màn hình chính

Gởi thông báo thoát khỏi Group sổ ¬ . : n:

Kiêm tra người dùng có tạo ra group này hay không

W Kiểm tra người tạo group

Kiểm tra đúng [Ƒ

Lấy danh sách các Users có trong Group

Danh sách các Users có trọng Group TLỊ

Thông báo hủy Group

Xóa GroupDetail Xóa Group

Hình 3-11: Mô hình xử lý hủy Group

> Mô tả:

Khi một người dùng tắt Tab Group Chat, Client sẽ gởi thông báo thoát khỏi Group cho Server. Server trước tiên kiểm tra người dùng này có phải là người đã tạo Group hay không. Nếu đúng, Server sẽ gởi thông báo hủy Group tới các Users khác trong Group. Cuối cùng, Server sẽ xóa Group này khỏi GroupDetail và GroupChat.

»> Ghi chú:

Khi nhận được thông báo hủy Group từ Users, Client sẽ tắt Tab Group Chat tương ứng đồng thời thông báo cho người dùng.

+ Khi User gởi tin nhắn tới các thành viên trong Group:

> Mô hình xửlý:

37

Gởi tin nhắn trong Group

KSà [ Ị (3

\ã@? \s7

+ Màn hình chính Senwer Table GroupDetail :

MÀ | MÀ

Ngưởi gởi Các Users trong Group

Nhập tin nhắn

Chọn Tab Group Chat và yêu cầu chức năng gời tin nhắn

Hiển thị tin nhắn

Gởi tin nhắn kèm theo tên Group

Lắầy danh sách các Usrs trong Group Danh sách các Users trong Group |

Gởi tin nhắn kèm theo tên người gời đến các Usrs trong Group

Hình 3-12: Mô hình xửlý gởi tin nhắn trong Group

> Mô tả:

Khi người dùng nhập tin nhắn và nhấn button Send trong Tab Group Chat, Client sẽ gởi tin nhắn và tên Group cho Server. Dựa vào tên Group nhận được, Server sẽ tìm tất cả các Users đã tham gia Group và gởi tin nhắn kèm theo tên người gởi đến các Users này.

* 1. Thiết kế dữ liệu

##### Chuẩn hóa dữ liệu:

> Users (TD, Ủsername, Password, State, Fullname, Email, Address)

> FriendLIst(UserID, FriendID)

> OfflineMessage(UserID, SenderID, SendTime, Message)

> GroupChat(GroupIDÐ. DserID, GroupName, CreateTime)

> GroupDetail(FriendID, GroupID)

38

* + 1. Mô hình dữ liệu ở mức vật lý:

Users

ID imt <ee

FK\_FRIENDLI\_FRIENDLIS\_USERS Username\_ varchar(20)

— ...- — #=“ Password varchar(18)

FriendList State bit

UserID int <pkfk1>

\_—\_— TU nvarchar(50)

EriendlD int <pkfk2> Email varchar(50)

FK\_FRIENDLI\_FRIENDLIS\_USERS „ Address nvarchar(50)

FK\_OFFLINEM\_OFFLINEME\_USERS \_“

»y FK\_GROUPDET\_GROUPDETA\_USERS

GroupDetail

FK\_GROUPCHA\_RELATIONS\_USERS

ErendID int GrouplÐ ¡in

<pkfk1>

<pkfk2>

OfflineMessage

FK\_OFFLIREM\_OFFLINEME\_USERS

FK\_GROUPDET\_GRỐUPDETA\_GROUPCHA

UserlD jnt <pkfk1>

SenderD ¡int <pk,fk2>

7n.

SendTime datetime | = GroupChat

Message nvarchar(500) GroupID int <pk. UserlD int <fc GroupName\_ varchar(30) CreateTime datetime

Hình 3-13: Mô hình dữ liệu ở mức vật lý

##### Thiết kế dữ liệu:

Gồm có các bảng Users, FriendList, OfflineMessage, GroupChat, GroupDetail để lưu trử các thông tin vê user đê đăng nhập, quan hệ giữa các user, lưu tin nhăn offline của các user và lưu tạm thời thông tin các nhóm Chat.

###### + Table Users

> Chức năng: Lưu thông tin về User

> Danh sách các thuộc tính:

Bảng 3-1: Table Users

##### STT Tên thuộc tính Kiểu dữ liệu Diễn giải

1 ID 1nt (4) Tự động tăng, Khóa chính

s Ủsername varchar (20) Tên đăng nhập

1. Password varchar (15) Mật khâu đăng nhập

1. Sfate bit (1) Trạng thái của Ủser

1. Fullname nvarchar (50) Tên đầy đủ của User

1. Email varchar (30) Email của Ủser

1. Address nvarchar (50) Địa chỉ của User

39

+ Table FriendList

> Chức năng: Lưu thông tin về quan hệ giữa các User

>\_ Danh sách các thuộc tính:

Bảng 3-2: Table FriendList

##### STT Tên thuộc tính Kiểu dữ liệu Diễn giải

1. UserID 1nt (4) ID của User, Khóa chính

1. FriendID Inf (4) ID của Friend, Khóa chính

###### + Table OfflineMessage

> Chức năng: Lưu các tin nhắn offline của User

> Danh sách các thuộc tính:

Bảng 3-3: Table OfflineMessage

##### STT Tên thuộc tính Kiểu dữ liệu Diễn giải

1 UserID Inf (4) ID của người nhận, Khóa chính

s5 SenderID Int (4) ID của người gởi, Khóa chính

##### 3 SendTime dateime(8) |Thời điểm gởi tin nhắn, Khóa

chính

4 Message nvarchar (500) Nội dung tin nhắn

4 Message nvarchar (500) Nội dung tin nhắn

###### 4+ Table GroupChat

> Chức năng: Lưu tạm thời thông tin của các nhóm Chat

> Danh sách các thuộc tính:

Bảng 3-4: Table GroupChat

##### STT Tên thuộc tính Kiểu dữ liệu Diễn giải

1 GroupID 1nt (4) Tự động tăng, Khóa chính

s UserID Int (4) ID của ỦUser tạo nhóm

1. GroupName varchar (20) Tên của nhóm

1. CreateTime datetime (8) Thời đim tạo nhóm

###### + Table GroupDetail

> Chức năng: Lưu thông tin chỉtiết của các nhóm Chat

> Danh sách các thuộc tính:

40

Bảng 3-5: Table GroupDetail

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuộc tính | Kiểu dữ liệu | Diễn giải |
| 1 GroupID 1nt (4) ID của Group, Khóa chính | | | |

ID của của Ủser thành viên trong

s FriendID Int (4)

Group, Khóa chính

3.2.4. Mô tả các ràng buộc toàn vẹn:

% Ràng buộc I: Username trong Table Users không được trùng nhau.

+ Ràng buộc 2: UserID và FriendID trong Table FriendList phải tồn tại trong Table Users.

4% Ràng buộc 3: UsersID trong Table GroupChat phải tồn tại trong Table Users.

+ Ràng buộc 4: GroupID trong Table GroupDetail phải tồn tại trong Table GroupChat.

& Ràng buộc 5: FriendlD trong Table GroupDetail phải tồn tại trong Table Users.

+ Ràng buộc 6: SenderID và UserlD trong Table OfflineMessage phải tồn tại trong Table Users.

###### Thiết kế giao diện

* + 1. Màn hình đăng nhập

##### s-} Chat withmẹ.

eIver |P 127001 Llserrnaine

Fas=worr

| Sinn in | | Eiit |

Hình 3-14: Màn hình đăng nhập

4I

* + 1. Màn hình chính

###### )wChatithme

File ÁcHons

SeF/er |

Hình 3-15: Màn hình chính

* + 1. Màn hình thêm Friend

Friend Account

Hình 3-16: Màn hình thêm Friend

42

* + 1. Màn hình xóa Friend

Friantl ÀÄccounL

Hình 3-17: Màn hình xóa Friend

.Account

Hình 3-18: Màn hình Chat With

43

3.3.6

Màn hình

Invite

Group

mẽ

People To

Diainricr

Frierids List Invitian Liat

†easter xelcn Tuxedo

lnvite Messaqe

Hình 3-19: Màn hình Invite Group

3.3.7 Màn hình Invite Another

Pennle

` Hình 3-20: Màn hình Invite Another

# Chương 4: CÀI ĐẶT- THỨ NGHIỆM

###### Cài đặt chương trình

* 1. Hướng dẫn sử dụng

Chương 5: KẾT LUẬN

* 1. Kết quả đạt được

4 Đề tài “Chương trình Chat” đã thực hiện được các nội dung sau:

44

> Tìm hiểu được cách thức lập trình Socket và lập trình đa luồng trên

môi trường .NET

> Tìm hiểu được cách thức hoạt động của một chương trình Chat đơn

giản.

##### + Ưu và nhược điểm của chương trình:

###### > Ưu điểm:

> Nhược điểm:

". Chưa gởi kèm các hình ảnh khi Chat.

“. Chưa có chức năng gởi File giữa các Users.

" Chihỗ trợ Chat Text đơn thuần.

###### Hướng phát triển

Về cơ bản, nghiên cứu đã đạt được những yêu cầu đã đặt ra. Tuy nhên nếu có điều kiện, đồ án sẽ cố gắng phát triển thêm các chức năng sau:

##### + Hỗ trợ gởi hình ảnh khi Chat

###### + Hỗtrợ chức năng gởifile.

+ Hỗ trợ chức năng Voice Chat và Webcam.

45

# TÀI LIỆU THAM KHẢO

Tài liệu:

1. Dương Quang Thiện, 2005, .NET toàn tập.
2. Nguyễn Ngọc Bình Phương, 2005, Các giải pháp lập trình Cứ.
3. Sybex, 2003, C-Sharp Network Programming Các trang Web
4. [http:/www.codeproJecf.com](http://www.codeproJecf.com/)
5. [http:/www.codeguru.com](http://www.codeguru.com/)

### 46