C#：一個增強的TcpClient（一）連接

sing System;

using System.Net;

using System.Net.Sockets;

using System.Threading;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 非同步連接狀態物件*

*/// </summary>*

internal class AsyncConnectStateObject

{

public ManualResetEvent eventDone;

public TcpClient client;

public Exception exception;

}

*/// <summary>*

*/// 實現TcpClient的非同步連接*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 設置連接逾時值*

*/// </summary>*

public Int32 ConnectTimeout = Timeout.Infinite;

*/// <summary>*

*/// 非同步連接*

*/// </summary>*

*/// <param name="hostname">主機名稱</param>*

*/// <param name="port">埠號</param>*

public void AsyncConnect(String hostname, Int32 port)

{

*// 使用者定義物件*

AsyncConnectStateObject State = new AsyncConnectStateObject

{ *// 將事件狀態設置為非終止狀態，導致執行緒阻止*

eventDone = new ManualResetEvent(false),

client = this,

exception = null

};

*// 開始一個對遠端主機連接的非同步請求*

BeginConnect(hostname, port, new AsyncCallback(AsyncConnectCallback), State);

*// 等待操作完成信號*

if (State.eventDone.WaitOne(ConnectTimeout, false))

{ *// 接收到信號*

if (State.exception != null) throw State.exception;

}

else

{ *// 超時異常*

Close();

throw new TimeoutException();

}

}

*/// <summary>*

*/// 非同步連接*

*/// </summary>*

*/// <param name="address">IP地址</param>*

*/// <param name="port">埠號</param>*

public void AsyncConnect(IPAddress address, Int32 port)

{

*// 使用者定義物件*

AsyncConnectStateObject State = new AsyncConnectStateObject

{ *// 將事件狀態設置為非終止狀態，導致執行緒阻止*

eventDone = new ManualResetEvent(false),

client = this,

exception = null

};

*// 開始一個對遠端主機連接的非同步請求*

BeginConnect(address, port, new AsyncCallback(AsyncConnectCallback), State);

*// 等待操作完成信號*

if (State.eventDone.WaitOne(ConnectTimeout, false))

{ *// 接收到信號*

if (State.exception != null) throw State.exception;

}

else

{ *// 超時異常*

Close();

throw new TimeoutException();

}

}

*/// <summary>*

*/// 非同步連接回呼函數*

*/// </summary>*

*/// <param name="ar">非同步作業結果</param>*

private static void AsyncConnectCallback(IAsyncResult ar)

{

AsyncConnectStateObject State = ar.AsyncState as AsyncConnectStateObject;

try

{ *// 非同步接受傳入的連接嘗試*

State.client.EndConnect(ar);

}

catch (Exception e)

{ *// 非同步連接異常*

State.exception = e;

}

finally

{ *// 將事件狀態設置為終止狀態，執行緒繼續*

State.eventDone.Set();

}

}

}

}

C#：一個增強的TcpClient（二）寫入

using System;

using System.Net.Sockets;

using System.Text;

using System.Threading;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 非同步寫狀態物件*

*/// </summary>*

internal class AsyncWriteStateObject

{

public ManualResetEvent eventDone;

public NetworkStream stream;

public Exception exception;

}

*/// <summary>*

*/// 實現TcpClient的非同步發送*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 設置寫超時等待值*

*/// </summary>*

public Int32 WriteTimeout = Timeout.Infinite;

*/// <summary>*

*/// 非同步發送*

*/// </summary>*

*/// <param name="buffer">位元組陣列</param>*

*/// <param name="offset">起始偏移量</param>*

*/// <param name="size">位元組數</param>*

public void Write(Byte[] buffer, Int32 offset, Int32 size)

{

*// 獲取網路資料流程*

NetworkStream netStream = GetStream();

*// 使用者定義物件*

AsyncWriteStateObject State = new AsyncWriteStateObject

{ *// 將事件狀態設置為非終止狀態，導致執行緒阻止*

eventDone = new ManualResetEvent(false),

stream = netStream,

exception = null

};

Byte[] BytesArray;

if (String.IsNullOrEmpty(\_secretKey))

{ *// 在資料前插入長度資訊*

Int32 Length = size + 4; *// 加入4位元組長度資訊後的總長度*

BytesArray = new Byte[Length];

Array.Copy(BitConverter.GetBytes(Length), BytesArray, 4);

Array.Copy(buffer, offset, BytesArray, 4, size);

}

else

{ *// 資料加密*

Byte[] Cipher = Encrypt(buffer, offset, size);

*// 在資料前插入長度資訊*

Int32 Length = Cipher.Length + 4;

BytesArray = new Byte[Length];

Array.Copy(BitConverter.GetBytes(Length), BytesArray, 4);

Array.Copy(Cipher, 0, BytesArray, 4, Cipher.Length);

}

*// 寫入加長度資訊頭的資料*

netStream.BeginWrite(BytesArray, 0, BytesArray.Length, new AsyncCallback(AsyncWriteCallback), State);

*// 等待操作完成信號*

if (State.eventDone.WaitOne(WriteTimeout, false))

{ *// 接收到信號*

if (State.exception != null) throw State.exception;

}

else

{ *// 超時異常*

throw new TimeoutException();

}

}

*/// <summary>*

*/// 非同步發送*

*/// </summary>*

*/// <param name="data">位元組陣列</param>*

public void Write(Byte[] data)

{

Write(data, 0, data.Length);

}

*/// <summary>*

*/// 非同步發送*

*/// </summary>*

*/// <param name="command">字串</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 936：簡體中文GB2312*

*/// 54936：簡體中文GB18030*

*/// 950：繁體中文BIG5*

*/// 1252：西歐字元CP1252*

*/// 65001：UTF-8編碼*

*/// </remarks>*

public void Write(String command, Int32 codePage = 65001)

{

Write(Encoding.GetEncoding(codePage).GetBytes(command));

}

*/// <summary>*

*/// 非同步寫入回呼函數*

*/// </summary>*

*/// <param name="ar">非同步作業結果</param>*

private static void AsyncWriteCallback(IAsyncResult ar)

{

AsyncWriteStateObject State = ar.AsyncState as AsyncWriteStateObject;

try

{ *// 非同步寫入結束*

State.stream.EndWrite(ar);

}

catch (Exception e)

{ *// 非同步連接異常*

State.exception = e;

}

finally

{ *// 將事件狀態設置為終止狀態，執行緒繼續*

State.eventDone.Set();

}

}

}

}

# C#：一個增強的TcpClient（三）讀取

using System;

using System.IO;

using System.Net.Sockets;

using System.Text;

using System.Threading;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 非同步讀狀態物件*

*/// </summary>*

internal class AsyncReadStateObject

{

public ManualResetEvent eventDone;

public NetworkStream stream;

public Exception exception;

public Int32 numberOfBytesRead;

}

*/// <summary>*

*/// 實現TcpClient的非同步接收*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 設置讀超時等待值*

*/// </summary>*

public Int32 ReadTimeout = Timeout.Infinite;

*/// <summary>*

*/// 非同步接收*

*/// </summary>*

*/// <param name="data">接收到的位元組陣列</param>*

public void Read(out Byte[] data)

{

*// 獲取網路資料流程*

NetworkStream netStream = GetStream();

*// 使用者定義物件*

AsyncReadStateObject State = new AsyncReadStateObject

{ *// 將事件狀態設置為非終止狀態，導致執行緒阻止*

eventDone = new ManualResetEvent(false),

stream = netStream,

exception = null,

numberOfBytesRead = -1

};

Byte[] Buffer = new Byte[ReceiveBufferSize];

using (MemoryStream memStream = new MemoryStream(ReceiveBufferSize))

{

Int32 TotalBytes = 0; *// 總共需要接收的位元組數*

Int32 ReceivedBytes = 0; *// 當前已接收的位元組數*

while (true)

{

*// 將事件狀態設置為非終止狀態，導致執行緒阻止*

State.eventDone.Reset();

*// 非同步讀取網路資料流程*

netStream.BeginRead(Buffer, 0, Buffer.Length, new AsyncCallback(AsyncReadCallback), State);

*// 等待操作完成信號*

if (State.eventDone.WaitOne(ReadTimeout, false))

{ *// 接收到信號*

if (State.exception != null) throw State.exception;

if (State.numberOfBytesRead == 0)

{ *// 連接已經斷開*

throw new SocketException();

}

else if (State.numberOfBytesRead > 0)

{

if (TotalBytes == 0)

{ *// 提取流頭部位元組長度資訊*

TotalBytes = BitConverter.ToInt32(Buffer, 0);

*// 保存剩餘資訊*

memStream.Write(Buffer, 4, State.numberOfBytesRead - 4);

}

else

{

memStream.Write(Buffer, 0, State.numberOfBytesRead);

}

ReceivedBytes += State.numberOfBytesRead;

if (ReceivedBytes >= TotalBytes) break;

}

}

else

{ *// 超時異常*

throw new TimeoutException();

}

}

*// 將流內容寫入位元組陣列*

if (String.IsNullOrEmpty(\_secretKey))

{

data = (memStream.Length > 0) ? memStream.ToArray() : null;

}

else

{ *// 解密數據*

data = (memStream.Length > 0) ? Decrypt(memStream.ToArray(), 0, TotalBytes - 4) : null;

}

}

}

*/// <summary>*

*/// 非同步接收*

*/// </summary>*

*/// <param name="answer">接收到的字串</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 936：簡體中文GB2312*

*/// 54936：簡體中文GB18030*

*/// 950：繁體中文BIG5*

*/// 1252：西歐字元CP1252*

*/// 65001：UTF-8編碼*

*/// </remarks>*

public void Read(out String answer, Int32 codePage = 65001)

{

Byte[] data;

Read(out data);

answer = (data != null) ? Encoding.GetEncoding(codePage).GetString(data) : null;

}

*/// <summary>*

*/// 非同步讀取回呼函數*

*/// </summary>*

*/// <param name="ar">非同步作業結果</param>*

private static void AsyncReadCallback(IAsyncResult ar)

{

AsyncReadStateObject State = ar.AsyncState as AsyncReadStateObject;

try

{ *// 非同步寫入結束*

State.numberOfBytesRead = State.stream.EndRead(ar);

}

catch (Exception e)

{ *// 非同步連接異常*

State.exception = e;

}

finally

{ *// 將事件狀態設置為終止狀態，執行緒繼續*

State.eventDone.Set();

}

}

}

}

# C#：一個增強的TcpClient（四）查詢

using System;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 實現TcpClient的非同步查詢*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="offset">起始偏移量</param>*

*/// <param name="size">位元組數</param>*

*/// <param name="answer">接收資料</param>*

public void Query(Byte[] command, Int32 offset, Int32 size, out Byte[] answer)

{

if (command != null)

{ *// 發送資料*

Write(command, offset, size);

}

*// 接收資料*

Read(out answer);

}

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="offset">起始偏移量</param>*

*/// <param name="size">位元組數</param>*

*/// <param name="answer">接收資料</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 簡體中文GB2312 936*

*/// 簡體中文GB18030 54936*

*/// 繁體中文BIG5 950*

*/// 西歐字元CP1252 1252*

*/// UTF-8 65001*

*/// </remarks>*

public void Query(Byte[] command, Int32 offset, Int32 size, out String answer, Int32 codePage = 65001)

{

if (command != null)

{ *// 發送資料*

Write(command, offset, size);

}

*// 接收資料*

Read(out answer, codePage);

}

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="answer">接收資料</param>*

public void Query(Byte[] command, out Byte[] answer)

{

if (command != null)

{ *// 發送資料*

Write(command);

}

*// 接收資料*

Read(out answer);

}

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="answer">接收資料</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 簡體中文GB2312 936*

*/// 簡體中文GB18030 54936*

*/// 繁體中文BIG5 950*

*/// 西歐字元CP1252 1252*

*/// UTF-8 65001*

*/// </remarks>*

public void Query(Byte[] command, out String answer, Int32 codePage = 65001)

{

if (command != null)

{ *// 發送資料*

Write(command);

}

*// 接收資料*

Read(out answer, codePage);

}

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="answer">接收資料</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 簡體中文GB2312 936*

*/// 簡體中文GB18030 54936*

*/// 繁體中文BIG5 950*

*/// 西歐字元CP1252 1252*

*/// UTF-8 65001*

*/// </remarks>*

public void Query(String command, out Byte[] answer, Int32 codePage = 65001)

{

if (!String.IsNullOrEmpty(command))

{ *// 發送資料*

Write(command, codePage);

}

*// 接收資料*

Read(out answer);

}

*/// <summary>*

*/// 非同步查詢*

*/// </summary>*

*/// <param name="command">發送資料</param>*

*/// <param name="answer">接收資料</param>*

*/// <param name="codePage">內碼表</param>*

*/// <remarks>*

*/// 內碼表：*

*/// 簡體中文GB2312 936*

*/// 簡體中文GB18030 54936*

*/// 繁體中文BIG5 950*

*/// 西歐字元CP1252 1252*

*/// UTF-8 65001*

*/// </remarks>*

public void Query(String command, out String answer, Int32 codePage = 65001)

{

if (!String.IsNullOrEmpty(command))

{ *// 發送資料*

Write(command, codePage);

}

*// 接收資料*

Read(out answer, codePage);

}

}

}

# C#：一個增強的TcpClient（五）構造函數

*/\* ----------------------------------------------------------*

*檔案名稱：TcpClientPlus.cs*

*作者：秦建輝*

*MSN：splashcn@msn.com*

*QQ：36748897*

*博客：http://blog.csdn.net/jhqin*

*開發環境：*

*Visual Studio V2010*

*.NET Framework 4 Client Profile*

*版本歷史：*

*V1.4 2012年04月24日*

*提供獨立的通信執行緒*

*V1.3 2012年04月19日*

*提供AES 256位資料加密傳輸功能*

*V1.2 2012年03月30日*

*在流頭部增加位元組長度資訊以方便讀寫*

*V1.1 2012年03月27日*

*增加非同步讀寫，改進軟體架構*

*V1.0 2012年03月16日*

*實現TcpClient的非同步連接（基於同事趙小亮的代碼優化）*

*調用順序：*

*1.產生實體，視需要分別設置ConnectTimeout、ReadTimeout、WriteTimeout欄位*

*2.如果未連接，則調用AsyncConnect建立連接*

*3.如果傳輸加密，設置SecretKey和Salt*

*4.重複調用ThreadTaskAllocation分發通信任務*

*5.調用Close關閉連接*

*------------------------------------------------------------ \*/*

using System;

using System.Net;

using System.Net.Sockets;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 構造函數和析構函數*

*/// </summary>*

public partial class TcpClientPlus : TcpClient

{

*/// <summary>*

*/// 構造函數*

*/// </summary>*

public TcpClientPlus() : base() { }

*/// <summary>*

*/// 構造函數*

*/// </summary>*

*/// <param name="family">IP地址的地址族</param>*

public TcpClientPlus(AddressFamily family) : base(family) { }

*/// <summary>*

*/// 構造函數*

*/// </summary>*

*/// <param name="localEP">將網路端點表示為 IP 位址和埠號</param>*

public TcpClientPlus(IPEndPoint localEP) : base(localEP) { }

*/// <summary>*

*/// 構造函數*

*/// </summary>*

*/// <param name="address">主機名稱或者IP地址</param>*

*/// <param name="port">埠號</param>*

public TcpClientPlus(String address, Int32 port)

: base()

{

*// 判斷address是主機名稱還是IP地址*

try

{ *// IPv4 使用以點分隔的四部分標記法，IPv6 使用冒號十六進位標記法*

IPAddress ip = IPAddress.Parse(address);

*// 遠端主機由IP位址和埠號指定*

Connect(ip, port);

}

catch (FormatException)

{

*// 遠端主機由主機名稱和埠號指定*

Connect(address, port);

}

}

*/// <summary>*

*/// 釋放資源*

*/// </summary>*

*/// <param name="disposing">*

*/// true：釋放託管資源和非託管資源*

*/// false：僅釋放非託管資源*

*/// </param>*

protected override void Dispose(bool disposing)

{

*// 終止獨立的通信執行緒*

ThreadTaskAbort();

*// 關閉加密傳輸模組*

SecurityClose();

*// 調用基類函數釋放資源*

base.Dispose(disposing);

}

*/// <summary>*

*/// 析構函數*

*/// </summary>*

~TcpClientPlus()

{ *// 僅釋放非託管資源*

Dispose(false);

}

}

}

# C#：一個增強的TcpClient（六）資料加密

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 實現傳送資料的加解密*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 雜湊演算法的金鑰*

*/// </summary>*

private const String \_hmackey = "昨夜星辰昨夜風，畫樓西畔桂堂東。身無彩鳳雙飛翼，心有靈犀一點通。";

*/// <summary>*

*/// AES對稱演算法的託管實現*

*/// </summary>*

protected AesManaged \_aes;

*/// <summary>*

*/// 加密金鑰欄位*

*/// </summary>*

private String \_secretKey;

*/// <summary>*

*/// 加密金鑰屬性*

*/// </summary>*

public String SecretKey

{

get { return \_secretKey; }

set

{

\_secretKey = value;

if (String.IsNullOrEmpty(value))

{ *// 關閉加密傳輸模組*

SecurityClose();

}

else

{

if (\_aes == null) \_aes = new AesManaged();

*// 更新加密金鑰（256位）*

using (SHA256Managed sha = new SHA256Managed())

{

\_aes.Key = sha.ComputeHash(Encoding.UTF8.GetBytes(value + \_salt));

sha.Clear(); *// 清除敏感性資料*

}

*// 更新初始向量（128位）*

using (HMACMD5 hmacmd5 = new HMACMD5(Encoding.UTF8.GetBytes(\_hmackey)))

{

\_aes.IV = hmacmd5.ComputeHash(Encoding.UTF8.GetBytes(\_salt));

hmacmd5.Clear(); *// 清除敏感性資料*

}

}

}

}

*/// <summary>*

*/// 最短密碼佐料長度*

*/// </summary>*

public const Int32 MinSaltLength = 16;

*/// <summary>*

*/// 密碼佐料欄位*

*/// </summary>*

private String \_salt = "雄關漫道真如鐵，而今邁步從頭越。從頭越，蒼山如海，殘陽如血。";

*/// <summary>*

*/// 密碼佐料屬性*

*/// </summary>*

public String Salt

{

get { return \_salt; }

set

{ *// 要求Salt的長度大於16個字元*

if (!String.IsNullOrEmpty(value) && value.Length >= MinSaltLength)

{

\_salt = value;

if (!String.IsNullOrEmpty(\_secretKey))

{ *// 更新加密金鑰（256位）*

using (SHA256Managed sha = new SHA256Managed())

{

\_aes.Key = sha.ComputeHash(Encoding.UTF8.GetBytes(\_secretKey + value));

sha.Clear(); *// 清除敏感性資料*

}

*// 更新初始向量（128位）*

using (HMACMD5 hmacmd5 = new HMACMD5(Encoding.UTF8.GetBytes(\_hmackey)))

{

\_aes.IV = hmacmd5.ComputeHash(Encoding.UTF8.GetBytes(value));

hmacmd5.Clear(); *// 清除敏感性資料*

}

}

}

}

}

*/// <summary>*

*/// 加密資料*

*/// </summary>*

*/// <param name="buffer">原始資料</param>*

*/// <param name="offset">位元組偏移量</param>*

*/// <param name="count">要寫入當前流的位元組數</param>*

*/// <returns>加密後的數據</returns>*

private Byte[] Encrypt(Byte[] buffer, Int32 offset, Int32 count)

{

using(MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, \_aes.CreateEncryptor(), CryptoStreamMode.Write))

{

cs.Write(buffer, offset, count);

cs.Close();

return ms.ToArray();

}

}

}

*/// <summary>*

*/// 解密數據*

*/// </summary>*

*/// <param name="buffer">原始資料</param>*

*/// <param name="offset">位元組偏移量</param>*

*/// <param name="count">要寫入當前流的位元組數</param>*

*/// <returns>解密後的數據</returns>*

private Byte[] Decrypt(Byte[] buffer, Int32 offset, Int32 count)

{

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, \_aes.CreateDecryptor(), CryptoStreamMode.Write))

{

cs.Write(buffer, offset, count);

cs.Close();

return ms.ToArray();

}

}

}

*/// <summary>*

*/// 關閉加密傳輸模組*

*/// </summary>*

private void SecurityClose()

{

if (\_aes != null)

{

\_aes.Clear(); *// 清除敏感性資料*

\_aes.Dispose(); *// 釋放資源*

\_aes = null;

}

}

}

}

# C#：一個增強的TcpClient（七）獨立通信執行緒

using System;

using System.Threading;

namespace Splash.Net.Sockets

{

*/// <summary>*

*/// 實現獨立的通信執行緒*

*/// </summary>*

public partial class TcpClientPlus

{

*/// <summary>*

*/// 通道空閒等待時間，預設1秒*

*/// </summary>*

private const Int32 IdleTimeout = 1000;

*// 委託聲明*

public delegate void ThreadTaskRequest(object sender, EventArgs e);

*// 定義一個委託類型的事件*

public event ThreadTaskRequest OnThreadTaskRequest;

*/// <summary>*

*/// 獨立的通信執行緒器*

*/// </summary>*

protected Thread \_TaskThread;

*/// <summary>*

*/// 通道空閒事件*

*/// </summary>*

protected ManualResetEvent \_ChannelIdleEvent;

*/// <summary>*

*/// 任務到達事件*

*/// </summary>*

protected ManualResetEvent \_TaskArrivedEvent;

*/// <summary>*

*/// 獨立通信執行緒結束信號*

*/// </summary>*

private volatile Boolean \_shouldStop;

*/// <summary>*

*/// 啟動獨立的通信執行緒*

*/// </summary>*

*/// <param name="action">執行緒任務處理函數</param>*

public void ThreadTaskStart()

{

if (\_TaskThread == null)

{

\_ChannelIdleEvent = new ManualResetEvent(true); *// 初始化通道空閒*

\_TaskArrivedEvent = new ManualResetEvent(false); *// 初始化任務空閒*

\_shouldStop = false;

*// 創建並啟動獨立的通信執行緒*

\_TaskThread = new Thread(new ThreadStart(ThreadTaskAction));

\_TaskThread.Start();

}

}

*/// <summary>*

*/// 終止獨立的通信執行緒*

*/// </summary>*

public void ThreadTaskAbort()

{ *// 終止獨立通信執行緒*

if (\_TaskThread != null)

{

\_shouldStop = true; *// 設置執行緒結束信號*

\_TaskArrivedEvent.Set(); *// 設置任務到達事件*

}

*// 關閉通道空閒事件*

if (\_ChannelIdleEvent != null)

{

\_ChannelIdleEvent.Close();

\_ChannelIdleEvent = null;

}

*// 關閉任務到達事件*

if (\_TaskArrivedEvent != null)

{

\_TaskArrivedEvent.Close();

\_TaskArrivedEvent = null;

}

}

*/// <summary>*

*/// 獨立通信執行緒任務派發*

*/// </summary>*

*/// <returns>*

*/// true：任務派發成功*

*/// false：任務派發失敗*

*/// </returns>*

*/// <remarks>*

*/// 執行OnThreadTaskRequest關聯的事件*

*/// </remarks>*

public Boolean ThreadTaskAllocation()

{ *// 啟動獨立的通信執行緒*

if (\_TaskThread == null)

{

ThreadTaskStart();

}

*// 等待通道空閒*

if (\_ChannelIdleEvent.WaitOne(IdleTimeout, false))

{

\_ChannelIdleEvent.Reset(); *// 設置通道忙*

\_TaskArrivedEvent.Set(); *// 設置任務到達*

return true; *// 任務派發成功*

}

else

{

return false; *// 任務派發失敗*

}

}

*/// <summary>*

*/// 獨立通信執行緒任務派發*

*/// </summary>*

*/// <param name="task">要派發的任務請求</param>*

*/// <returns>*

*/// true：任務派發成功*

*/// false：任務派發失敗*

*/// </returns>*

*/// <remarks>*

*/// 更新OnThreadTaskRequest為當前任務並執行*

*/// </remarks>*

public Boolean ThreadTaskAllocation(ThreadTaskRequest task)

{ *// 啟動獨立的通信執行緒*

if (\_TaskThread == null)

{

ThreadTaskStart();

}

*// 等待通道空閒*

if (\_ChannelIdleEvent.WaitOne(IdleTimeout, false))

{ *// 設置通道忙*

\_ChannelIdleEvent.Reset();

*// 清空事件調用列表*

if (OnThreadTaskRequest != null)

{

foreach (Delegate d in OnThreadTaskRequest.GetInvocationList())

{

OnThreadTaskRequest -= (ThreadTaskRequest)d;

}

}

*// 更新事件調用列表*

OnThreadTaskRequest += task;

*// 設置任務到達*

\_TaskArrivedEvent.Set();

return true; *// 任務派發成功*

}

else

{

return false; *// 任務派發失敗*

}

}

*/// <summary>*

*/// 獨立通信執行緒器*

*/// </summary>*

private void ThreadTaskAction()

{

try

{

while (true)

{ *// 等待任務到達*

if (\_TaskArrivedEvent.WaitOne())

{ *// 檢測執行緒結束信號*

if (\_shouldStop) break;

try

{ *// 執行任務*

if (OnThreadTaskRequest != null)

{

OnThreadTaskRequest(this, EventArgs.Empty);

}

}

catch

{

*// 阻止異常拋出*

}

*// 等待新的任務*

if (\_TaskArrivedEvent != null) \_TaskArrivedEvent.Reset();

*// 設置通道空閒*

if (\_ChannelIdleEvent != null) \_ChannelIdleEvent.Set();

*// 再次檢測執行緒結束信號*

if (\_shouldStop) break;

}

} *// End While*

}

catch

{

*// 阻止異常拋出*

}

*// 保證執行緒資源釋放*

finally

{ *// 執行緒關閉*

\_TaskThread = null;

*// 關閉通道空閒事件*

if (\_ChannelIdleEvent != null)

{

\_ChannelIdleEvent.Close();

\_ChannelIdleEvent = null;

}

*// 關閉任務到達事件*

if (\_TaskArrivedEvent != null)

{

\_TaskArrivedEvent.Close();

\_TaskArrivedEvent = null;

}

}

}

}

}

# C#：一個增強的TcpClient（\*）用戶端演示程式及原始程式碼下載

using System;

using System.Net.Sockets;

using System.Windows.Forms;

using Splash.Net.Sockets;

using Splash.Windows.Forms;

namespace RunClient

{

public partial class Form1 : Form

{

*/// <summary>*

*/// TCP連接用戶端*

*/// </summary>*

private TcpClientPlus tcpClient;

public Form1()

{

InitializeComponent();

}

*// 建立連接*

private void button\_Connect\_Click(object sender, EventArgs e)

{

if (tcpClient == null)

{

try

{ *// 建立Tcp連接*

tcpClient = new TcpClientPlus(textBox\_IP.Text, Convert.ToInt32(textBox\_Port.Text));

button\_Connect.Text = "斷開連接";

MessageBoxPlus.Show(this, "建立連接成功！", "信息");

}

catch

{

MessageBoxPlus.Show(this, "建立連接失敗！", "信息");

button\_Send.Enabled = false; *// 發送按鈕失效*

}

}

else

{

tcpClient.Close();

tcpClient = null;

button\_Connect.Text = "建立連接";

MessageBoxPlus.Show(this, "斷開連接成功！", "信息");

}

}

*// 派發任務*

private void button\_Send\_Click(object sender, EventArgs e)

{

if (tcpClient == null || !tcpClient.Connected)

{

MessageBoxPlus.Show(this, "連接已中斷，請重新建立連接！", "信息");

return;

}

*// 發送按鈕失效*

button\_Send.Enabled = false;

if (String.IsNullOrEmpty(textBox\_Command.Text)) return;

*// 設置通信金鑰*

tcpClient.SecretKey = textBox\_SecretKey.Text;

*// 派發任務*

if (!tcpClient.ThreadTaskAllocation(HandleClientComm))

{

MessageBoxPlus.Show(this, "通信通道忙！", "信息");

}

}

*// 清空列表*

private void button\_Clear\_Click(object sender, EventArgs e)

{

textBox\_Notes.Clear();

}

private void textBox\_Command\_TextChanged(object sender, EventArgs e)

{

if (tcpClient != null)

{

if (!String.IsNullOrEmpty(textBox\_Command.Text))

{

button\_Send.Enabled = true;

}

}

}

*// 表單關閉*

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (tcpClient != null)

{

tcpClient.Close();

}

}

*// 處理通信*

private void HandleClientComm(object sender, EventArgs e)

{

TcpClientPlus client = sender as TcpClientPlus;

if (client != null)

{

try

{

String Command = GetCommand();

String Answer;

client.Query(Command, out Answer);

SetNote(Command + "\r\n");

SetNote(Answer + "\r\n\r\n");

}

catch (Exception ex)

{

Type type = ex.GetType();

if (type == typeof(SocketException) || type == typeof(System.IO.IOException))

{ *// 連接中斷*

client.Close();

MessageBoxPlus.Show(this, "連接中斷！", "信息");

}

else

{

SetNote("操作失敗異常原因：" + type.Name + "\r\n\r\n");

}

}

}

}

*// 對 Windows 表單控制項進行執行緒安全調用*

private void SetNote(String text)

{

if (textBox\_Notes.InvokeRequired)

{

textBox\_Notes.BeginInvoke(new Action<String>((msg) =>

{

textBox\_Notes.AppendText(msg);

}), text);

}

else

{

textBox\_Notes.AppendText(text);

}

}

*// 對 Windows 表單控制項進行執行緒安全調用*

private String GetCommand()

{

if (textBox\_Command.InvokeRequired)

{

return (String)textBox\_Command.Invoke(new Func<String>(() => { return textBox\_Command.Text; }));

}

else

{

return textBox\_Command.Text;

}

}

}

}