新北洋logo

**.Net基础类库接口文档**

**编 制： 于飞飞**

**审 核： 曹孔强**

**批 准： 李德逸**

**受控状态： 受控**

版权说明

|  |
| --- |
| 本文件中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属山东新北洋有限公司所有，受到有关产权及版权法保护。  本文件及其附录的知识产权归山东新北洋科技有限公司所有。如无授权，请勿复制或传播。  本文件将交付山东新北洋使用，使用范围仅局限于山东新北洋。 |

历史版本信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **日期** | **编制** | **审核** | **批准** | **备注** |
| V1.0.0 | 2017-10-13 | 于飞飞 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

版本变更说明

|  |  |
| --- | --- |
| **当前版本号** | **变更内容** |
| V1.0.0 | 发布评审 |
|  |  |

目 录

[1 总体概述 6](#_Toc501978580)

[2 SocketUtils(通讯库) 6](#_Toc501978581)

[2.1 TcpClientHelper(TCP客户端类) 6](#_Toc501978582)

[2.1.1 构造函数 6](#_Toc501978583)

[2.1.2 收到数据之后的委托函数 6](#_Toc501978584)

[2.1.3 设置接收到数据时调用的委托回调函数 7](#_Toc501978585)

[2.1.4 设置数据域长度 7](#_Toc501978586)

[2.1.5 连接到远程主机 8](#_Toc501978587)

[2.1.6 发送数据 8](#_Toc501978588)

[2.1.7 主动接收函数 8](#_Toc501978589)

[2.1.8 关闭连接 9](#_Toc501978590)

[2.2 TcpServerHelper(TCP服务端类) 9](#_Toc501978591)

[2.2.1 构造函数 9](#_Toc501978592)

[2.2.2 收到数据之后的委托函数 9](#_Toc501978593)

[2.2.3 设置接收到数据时调用的委托回调函数 9](#_Toc501978594)

[2.2.4 设置数据域长度 10](#_Toc501978595)

[2.2.5 启动监听 11](#_Toc501978596)

[2.2.6 发送数据 11](#_Toc501978597)

[2.2.7 关闭连接 11](#_Toc501978598)

[2.3 UdpHelper(UDP类) 11](#_Toc501978599)

[2.3.1 构造函数 11](#_Toc501978600)

[2.3.2 收到数据之后的委托函数 12](#_Toc501978601)

[2.3.3 设置接收到数据时调用的委托回调函数 12](#_Toc501978602)

[2.3.4 发送数据 13](#_Toc501978603)

[2.3.5 关闭连接 13](#_Toc501978604)

# 总体概述

目前系研的主要项目都是基于Java开发、.Net开发，在项目开发过程中，有些公用的功能如数据库操作、xml操作、网络通讯等等，在一部分项目进行了公用，但代码都在各自的项目中进行管理，没有统一的发布、管理和维护。

所以我们需要将公用的功能提炼出来，输出Java和.Net方向的技术货架，产品开发和项目开发人员需要什么样的技术，可以从技术货架中即拿即用，并且可以得到相应示例和指导。

.net的基础类库共输出8个动态库。其中SocketUtils类库包含TCP、UDP通讯功能，本文档主要描述SocketUtils动态库的接口。

# SocketUtils(通讯库)

## TcpClientHelper(TCP客户端类)

注意：TCP传输采用的是缓冲区存储的方式，缓冲区满时请及时对数据进行处理。

### 构造函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public TcpClientHelper(string localId , int localPort) | | | |
| **函数描述** | 构造函数(自动分配本地地址) | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | localId | String | 本地IP(标识) |  |
| localPort | int | 本地端口(标识) |  |
| **返回值** | 无 | | | |

### 收到数据之后的委托函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public delegate void ReceiveDataControl(byte[] receiveData) | | | |
| **函数描述** | 收到数据之后的委托函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | receiveData | byte[] | 收到的数据 |  |
| **返回值** |  | | | |

见6.1.3后的函数示例

### 设置接收到数据时调用的委托回调函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool SetReceiveFunction(ReceiveDataControl recvFun) | | | |
| **函数描述** | 设置接收到数据时调用的委托回调函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | recvFun | ReceiveDataControl | ReceiveDataControl委托 |  |
| **返回值** | Bool:返回是否设置成功 | | | |

例如：

在客户端tesTcpClientHelper1连接到服务端成功后：

//定义委托，执行ReceiveAndShowData方法

TcpClientHelper.ReceiveDataControl receiveDataControl = new TcpServerHelper.ReceiveDataControl (ReceiveAndShowData);

//设置客户端实例调用委托

tesTcpClientHelper1.SetReceiveFunction(receiveDataControl);

/// <summary>

/// 在接收到信息时调用，并对接收到的信息进行处理

/// </summary>

/// <param name="receiveBytes">接收到的信息</param>

private void ReceiveAndShowData(byte[] receiveBytes)

{

}

或者：

//设置客户端实例直接调用委托

tesTcpClientHelper1.SetReceiveFunction(ReceiveAndShowData);

/// <summary>

/// 在接收到信息时调用，并对接收到的信息进行处理，参数receiveBytes

/// </summary>

/// <param name="receiveBytes">接收到的信息</param>

private void ReceiveAndShowData(byte[] receiveBytes)

{

}

### 设置数据域长度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool SetLengthDetail(int startPos, int len, int type,int code,int numberStyle) | | | |
| **函数描述** | 用于描述接收数据域长度 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | startPos | int | 长度标识 开始位置 |  |
| len | int | 数据域的长度标识占几位字节 范围1-4 |  |
| type | int | 数据长度标识表示类型：  1：长度域表示全部数据长度；  2：长度域表示后续长度(含自身)；  3：长度域表示后续长度(不含自身) |  |
| code | int | 编码方式：  1:ascii  2:bcd  3:realdata |  |
| numberStyle | int | 进制：  1:10进制  2:16进制 | 若code==3,  1：小端  2:大端 |
| **返回值** | Bool:返回是否设置成功 | | | |

### 连接到远程主机

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Connect(string peerIp, int peerPort) | | | |
| **函数描述** | 连接到远程主机 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | peerIp | String | 远程Ip |  |
| peerPort | String | 远程Port |  |
| **返回值** | Bool:返回是否连接成功[当服务端达到最大连接数时，虽然连接(Connect)成功但是不被服务端Accept，即不能进行数据通信，此时返回true，待再次发送数据时，会检测出无法发送数据] | | | |

### 发送数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Send(byte[] sendBytes) | | | |
| **函数描述** | 发送数据 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | sendBytes | byte[] | 待发送的数据 |  |
| **返回值** | Bool:返回是否发送成功 | | | |

### 主动接收函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Receive(out byte[] recvBytes,int nTimeOut) | | | |
| **函数描述** | 主动接收函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | recvBytes | byte[] | 收到数据 |  |
| nTimeOut | int | 超时(毫秒) |  |
| **返回值** | Bool:返回是否接收成功 | | | |

### 关闭连接

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Close() | | | |
| **函数描述** | 关闭连接 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **返回值** | Bool:返回是否关闭成功 | | | |

## TcpServerHelper(TCP服务端类)

注意：TCP传输采用的是缓冲区存储的方式，缓冲区满时请及时对数据进行处理。

### 构造函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public TcpServerHelper(string localIp, int localPort) | | | |
| **函数描述** | 构造函数，绑定IP和端口 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | localIp | String | 本地IP |  |
| localPort | String | 本地Port |  |
| **返回值** | 无 | | | |

### 收到数据之后的委托函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public delegate void ReceiveDataControl(byte[] receiveData, string peerIp, int peerPort) | | | |
| **函数描述** | 收到数据之后的委托函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | receiveData | byte[] | 收到的数据 |  |
| peerIp | string | 发送数据的客户端Ip |  |
| peerPort | int | 发送数据的客户端端口 |  |
| **返回值** |  | | | |

见6.2.3后的示例

### 设置接收到数据时调用的委托回调函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool SetReceiveFunction(ReceiveDataControl recvFun) | | | |
| **函数描述** | 设置接收到数据时调用的委托回调函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | recvFun | ReceiveDataControl | ReceiveDataControl委托 |  |
| **返回值** | Bool：返回是否设置成功 | | | |

例如：

服务端实例\_tcpServer在连接到一个客户端之后：

//定义委托，执行ReceiveData方法

TcpServerHelper.ReceiveDataControl receiveDataControl = new TcpServerHelper.ReceiveDataControl(ReceiveData);

//设置服务端实例调用委托

\_tcpServer.SetReceiveFunction(receiveDataControl);

/// <summary>

/// 接收到数据之后回调ReceiveData函数，处理IP、端口和收到的数据

/// </summary>

/// <param name="receiveData">接收到的数据</param>

/// <param name="peerIp">对方IP</param>

/// <param name="peerPort">对方Port</param>

public void ReceiveData(byte[] receiveData, string peerIp, int peerPort)

{

}

或者：

//设置服务端实例直接调用委托

\_tcpServer.SetReceiveFunction(ReceiveData);

/// <summary>

/// 接收到数据之后回调ReceiveData函数，处理IP、端口和收到的数据

/// </summary>

/// <param name="receiveData">接收到的数据</param>

/// <param name="peerIp">对方IP</param>

/// <param name="peerPort">对方Port</param>

public void ReceiveData(byte[] receiveData, string peerIp, int peerPort)

{

}

### 设置数据域长度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool SetLengthDetail(int startPos, int len, int type,int code,int numberStyle) | | | |
| **函数描述** | 用于描述接收数据域长度 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | startPos | int | 长度标识 开始位置 |  |
| len | int | 数据域的长度标识占几位字节 范围1-4 |  |
| type | int | 数据长度标识表示类型：  1：长度域表示全部数据长度；  2：长度域表示后续长度(含自身)；  3：长度域表示后续长度(不含自身) |  |
| code | int | 编码方式：  1:ascii  2:bcd  3:realdata |  |
| numberStyle | int | 进制：  1:10进制  2:16进制 | 若code==3,  1：小端  2:大端 |
| **返回值** | Bool：返回是否设置成功 | | | |

### 启动监听

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool StartListen(int maxConnection) | | | |
| **函数描述** | 连接到远程主机 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | maxConnection | int | 最大可accept[进行数据通讯]的客户端连接数 |  |
| **返回值** | Bool：监听是否启动成功 | | | |

### 发送数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Send(byte[] sendBytes, string peerIp, int peerPort) | | | |
| **函数描述** | 发送数据 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | sendBytes | byte[] | 待发送的数据 |  |
| peerIp | String | 远程IP |  |
| peerPort | String | 远程Port |  |
| **返回值** | Bool：返回是否发送成功 | | | |

### 关闭连接

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool Close(string peerIp = "") | | | |
| **函数描述** | 关闭连接 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | peerIp | String | 连接的客户端Ip或服务器IP |  |
| **返回值** | Bool：返回是否关闭成功 | | | |

## UdpHelper(UDP类)

### 构造函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public UdpHelper(string localIp, int localPort) | | | |
| **函数描述** | 构造函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | localIp | String | 本地IP |  |
| localPort | String | 本地Port |  |
| **返回值** | 无 | | | |

### 收到数据之后的委托函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public delegate void ReceiveDataControl(byte[] receiveData, string peerIp, int peerPort) | | | |
| **函数描述** | 收到数据之后的委托函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | receiveData | byte[] | 接收到的数据 |  |
| peerIp | String | 接受发送方IP |  |
| peerPort | int | 接收发送方的Port |  |
| **返回值** | 无 | | | |

### 设置接收到数据时调用的委托回调函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public bool SetReceiveFunction(ReceiveDataControl recvFun) | | | |
| **函数描述** | 设置接收到数据时调用的委托回调函数 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | recvFun | ReceiveDataControl | ReceiveDataControl委托 |  |
| **返回值** | Bool：返回是否设置成功 | | | |

例如：

UDP实例\_udp在连接到一个客户端之后：

//定义委托，执行ReceiveData方法

UdpHelper.ReceiveDataControl receiveDataControl = new UdpHelper.ReceiveDataControl(ReceiveData);

//设置服务端实例调用委托

\_udp.SetReceiveFunction(receiveDataControl);

/// <summary>

/// 接收到数据之后，回调ReceiveData函数，处理IP、端口和收到的数据

/// </summary>

/// <param name="receiveData">接收到的数据</param>

/// <param name="peerIp">对方IP</param>

/// <param name="peerPort">对方Port</param>

public void ReceiveData(byte[] receiveData, string peerIp, int peerPort)

{

}

或者：

//设置服务端实例直接调用委托

\_udp.SetReceiveFunction(ReceiveData);

/// <summary>

/// 接收到数据之后，回调ReceiveData函数，处理IP、端口和收到的数据

/// </summary>

/// <param name="receiveData">接收到的数据</param>

/// <param name="peerIp">对方IP</param>

/// <param name="peerPort">对方Port</param>

public void ReceiveData(byte[] receiveData, string peerIp, int peerPort)

{

}

### 发送数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public int Send(byte[] sendBytes, string peerIp, int peerPort) | | | |
| **函数描述** | 发送数据 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | sendBytes | byte[] | 待发送的数据 |  |
| peerIp | String | 远程IP |  |
| peerPort | String | 远程Port |  |
| **返回值** | Int:返回是否发送成功  1：成功；  -1：失败 | | | |

### 关闭连接

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名称** | public int Close() | | | |
| **函数描述** | 关闭连接 | | | |
| **参数列表** | **变量名** | **类型** | **变量含义** | **备 注** |
| **参数** | peerIp | String | 配对的Ip |  |
| **返回值** | Int:返回是否关闭成功  1:成功；  -1：失败 | | | |