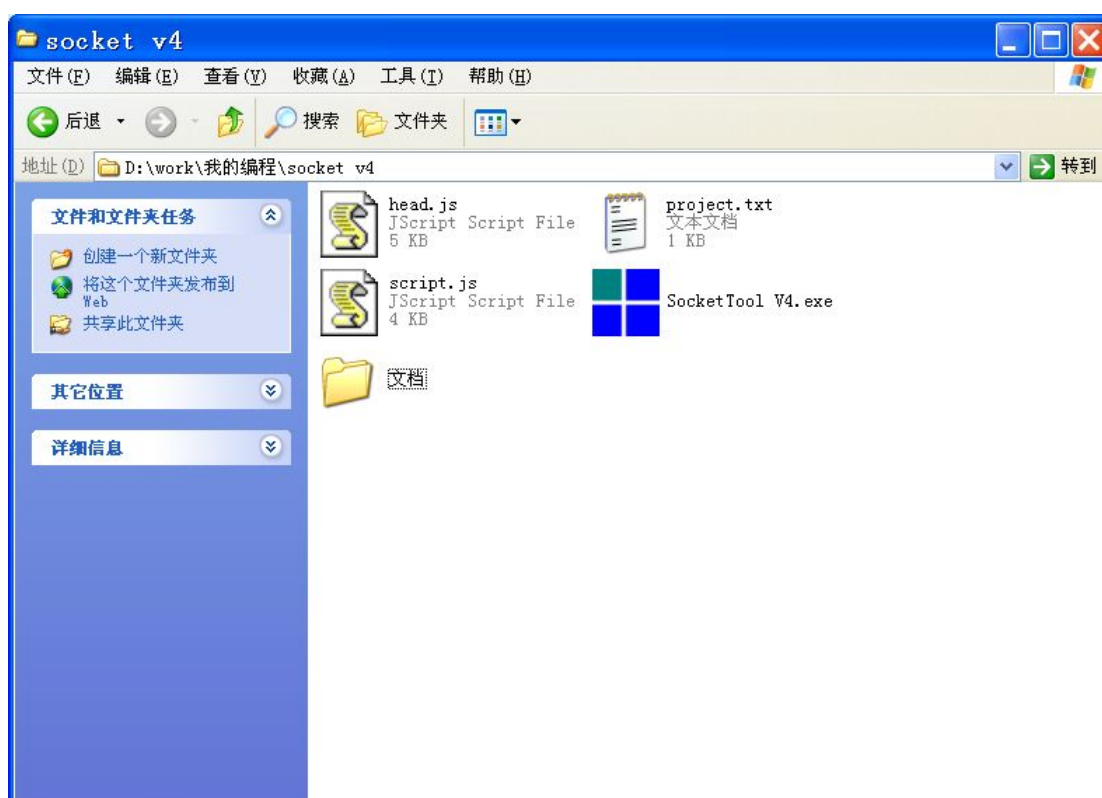


SocketTool V4.0 二次开发说明

一. 综述

SocketTool V4.0 版本支持 JavaScript 编程方式，从而增加了灵活性。

JavaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。



1) 工程文件

在目录下有一个工程文件：project.txt，这个文件必须处于 sockettool.exe 相同目录，并且文件名称不能改变。



在 project.txt 里面可以看到里面一共有两个 js 文件。一个是 head.js, 另一个是 script.js

其中 head.js 主要是底层的一些封装函数，使用者一般是不需要修改的。

Script.js 则主要用于用户编程来实现自己的功能。

用户还可以创建其他的.js 文件，比如一些.js 库文件等，包含在 Project 工程文件中，用户添加的其他文件应该放在在 script.js 前面，例如：



2) Json 对象说明

Sockettol javascript 采用了以下几个 Json 对象格式

A) socket 与 sockets

一个 socket 对象，代表一个套接字。 sockets 是全局套接字数组。

举例说明，声明一个 socket

```
var socket1={ID:"TcpServer01",SOCKETTYPE:"TCP Server",PORT:60000,HEX:1,UI:[]};
```

```
Sockets.push(socket1);
```

ID 是 socket 的名称，不能重复创建相同 ID 的 socket.

SOCKETTYPE 是 socket 的类型 , 分别有 TCP Server,TCP Client,UDP Server,UDP Client,UDP

Group 五种类型。

PORT 为端口号，范围是 1 - 65535

HEX：表示是否采用 HEX 格式显示收发的数据 0，1

UI 是用户交互元件组，用户可以在这里定义自己的文本框，按钮，标签文字。

sockets 是全局 socket 数组，已经在 head.js 内定义： var sockets=[];

socket 是支持循环创建的：

比如：

```
for(i=1;i<=3;i++){
  var socket3={ID:"TcpClient0"+i,SOCKETTYPE:"TCP Client",HOST:"127.0.0.1",PORT:60000,HEX:0,UI:[]};
  sockets.push(socket3);
}
```

就可以创建 3 个 TCP Client, 编号分别为：TcpClient01,TcpClient02,TcpClient03

B) action 与 actions

一个 action 代表一个执行动作，actions 是全局动作数组。

举例说明，声明一个 action：

```
var action={TYPE:"DISPLAY",DATA:text,SOCKET:socketid};
```

```
var actions.push(action);
```

TYPE 是动作类别的名称，分表有以下几个动作类别：

动作类别	含义	例子	说明
DISPLAY	在数据接收及提示窗口 增加一行显示	{TYPE:"DISPLAY",DATA: ' hello' ,SOCKET:socketid }	DATA 表示要显示的内容,socketid 对应 socket 编号

SEND	发送数据	{TYPE:"SEND",DATA:'313233',SOCKET:socketid}	DATA 表示要发送的内容，HEX 格式
CONNECT	建立 tcp 连接	{TYPE:"CONNECT",SOCKET:socketid}	只适用于 TCP Client 对象
DISCONNECT	断开 tcp 连接	{TYPE:"DISCONNECT",SOCKET:socketid}	适用于 TCP Client 对象和 TCP ServerClient 对象
ALERT	弹出提示框	{TYPE:"ALERT",TEXT:'hello',"SOCKET":socketid}	
ADDTIMER	创建定时器	{"TYPE":"ADDTIMER","TIMERNAME":timerid,"INTERVAL":interval,"SOCKET":socketid}	每个 socket 可以创建多个定时器,但定时器名称不能重复,定时器间隔以 10 毫秒为单位,定时器将产生定时器事件。
DELTIMER	删除定时器	{"TYPE":"DELTIMER","TIMERNAME":timerid,"SOCKET":socketid}	只能删除已经创建的定时器
SETTIMER	设置定时器	{"TYPE":"SETTIMER","TIMERNAME":timerid,"INTERVAL":interval}	重新设置已有定时器的时间间隔

		TERVAL":interval,"SOC KET":socketid}	
--	--	---	--

注：无论是数据的收发，如在 Send 中要发送的数据，或 OnSocketEvent 的 OnRead 中的数据，都是以 HEX 方式来表达的。

在 Head.js 中，内置了 toHex 和 toAscii 函数来进行转换。

如 Send(socketid, '010203'); 则实际上发送的是 0x01,0x02,0x03 三个字节。

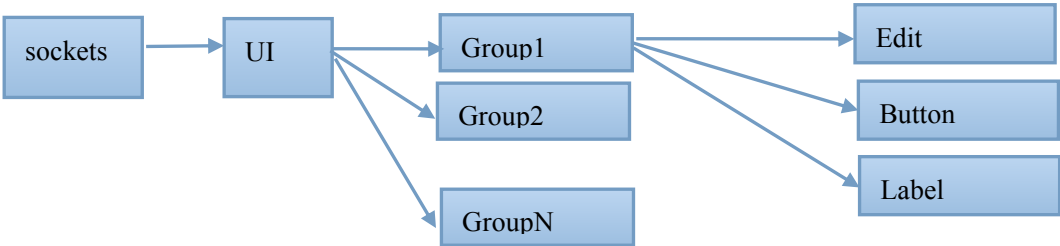
Send(socketid, toHex('hello!')); 则发送出去的是 hello! 6 个字符。

C) UI 人机交互元件对象：

在每个 socket 对象中，都有一个称为 UI 的数组，ui:[],

在 UI 数组内，包含多个 Group,每个 Group 对应一个工具条，

在每个 Group 里，又可以创建文本编辑框，按钮，和标签三种可视元件。



三种元件可以分别通过 AddLabel , AddButton,AddEdit 三个函数来创建。

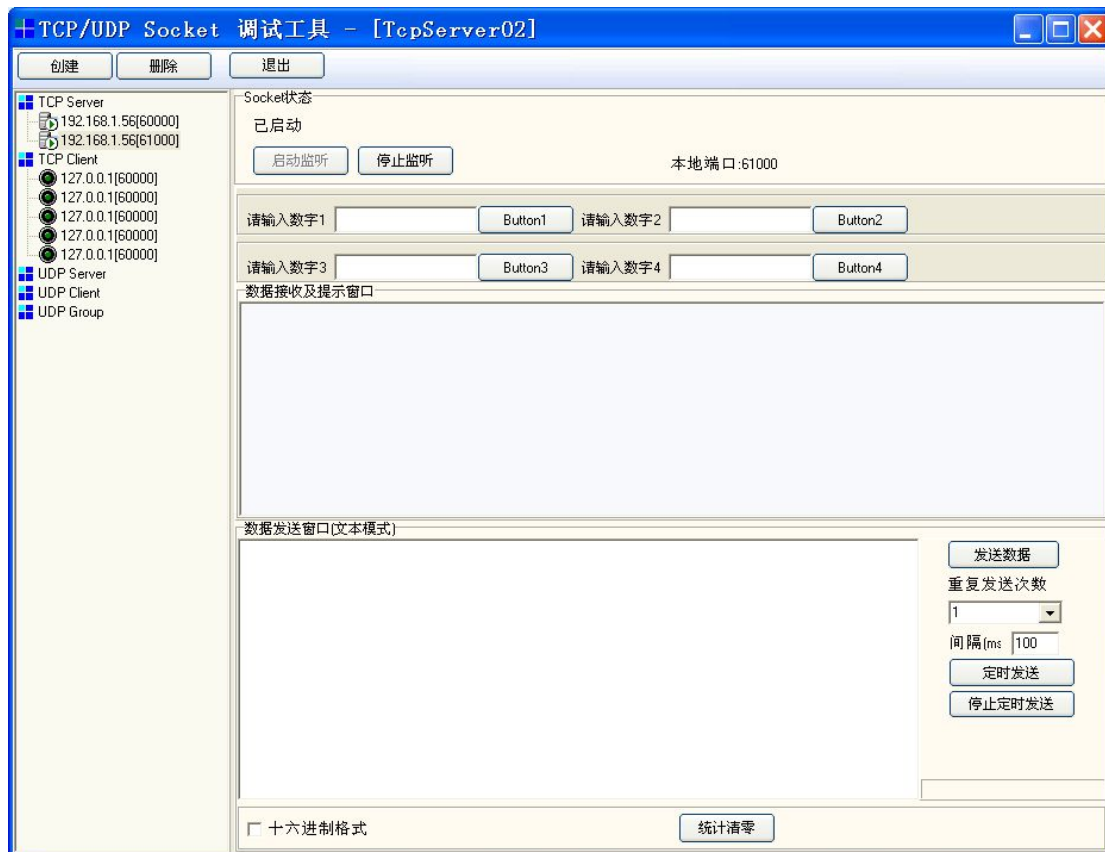
函数原型分别为：

function AddEdit(groupbox,name,text)

function AddButton(groupbox,name,text)

function AddLabel(groupbox,name,text,width)

例如：



我们可以看到名字为 TcpServer02 的 socket，在数据接收及提示窗口上方，有两个工具条，总共放置了 12 个元件，对应的 JavaScript 代码如下：

```
var group1=[];
```

```
AddLabel(group1,'Label1','请输入数字 1',80); AddEdit(group1,'Edit1','');
```

```
AddButton(group1,'Button1','Button1');
```

```
AddLabel(group1,'Label1','请输入数字 2',80); AddEdit(group1,'Edit2','');
```

```
AddButton(group1,'Button2','Button2');
```

```

var group2=[];

AddLabel(group2,'Label1','请输入数字 3',80);AddEdit(group2,'Edit3','');

AddButton(group2,'Button3','Button3');

AddLabel(group2,'Label1','请输入数字 4',80);AddEdit(group2,'Edit4','');

AddButton(group2,'Button4','Button4');


var socket2={ID:"TcpServer02",SOCKETTYPE:"TCP Server",PORT:61000,HEX:0,UI:[]};

socket2.UI.push(group1);

socket2.UI.push(group2);

sockets.push(socket2);

```

可视元件的数据读取和设置，则分别通过 GetUITex 和 SetUIText 函数来实现：

```

function SetUIText(socketid,uiid,text)

function GetUIText(socketid,uiid)

```

3) 系统事件说明

系统事件分为几个：

A) OnInit()

系统初始化事件，在 sockettool 启动时，会首先执行这个事件，通常我们在这个事件里面，创建我们主要创建的 socket，定时器，人机界面元件等：

例如：

```

function OnInit(){

```

```
var socket1={ID:"TcpServer01",SOCKETTYPE:"TCP Server",PORT:60000,HEX:1,UI:[]};
```

```
sockets.push(socket1);
```

```
var group1=[];
```

```
AddLabel(group1,'Label1','请输入数字 1',80); AddEdit(group1,'Edit1','');
```

```
AddButton(group1,'Button1','Button1');
```

```
AddLabel(group1,'Label1','请输入数字 2',80); AddEdit(group1,'Edit2','');
```

```
AddButton(group1,'Button2','Button2');
```

```
var group2=[];
```

```
AddLabel(group2,'Label1','请输入数字 3',80);AddEdit(group2,'Edit3','');
```

```
AddButton(group2,'Button3','Button3');
```

```
AddLabel(group2,'Label1','请输入数字 4',80);AddEdit(group2,'Edit4','');
```

```
AddButton(group2,'Button4','Button4');
```

```
var socket2={ID:"TcpServer02",SOCKETTYPE:"TCP Server",PORT:61000,HEX:0,UI:[]};
```

```
socket2.UI.push(group1);
```

```
socket2.UI.push(group2);
```

```
sockets.push(socket2);
```

```

for(i=1;i<=5;i++){

    var socket3={ID:"TcpClient"+i,SOCKETTYPE:"TCP
Client",HOST:"127.0.0.1",PORT:60000,HEX:0,UI:[]};

    var group3=[];

    AddEdit(group3,'Edit1',''); AddButton(group3,'Button1','Button1');

    AddEdit(group3,'Edit2',''); AddButton(group3,'Button2','Button2');

    socket3.UI.push(group3);

    sockets.push(socket3);

    AddTimer(socket3.ID,'timer01',10);

}

}

```

B) OnSocketEvent 事件

任何一个 socket 的连接，断开，以及收据收发，都会触发 OnSocketEvent 事件

```
function OnSocketEvent(socketid,eventname,param1,param2);
```

Socketid 是 socket 的编号，

eventname 对应事件名称，有 OnConnect，OnDisConnect,OnRead 三种

param1,param2 分别是该事件下的参数，比如 OnRead 时，param1 就是读取到的数据内容

B) OnButtonClick 事件

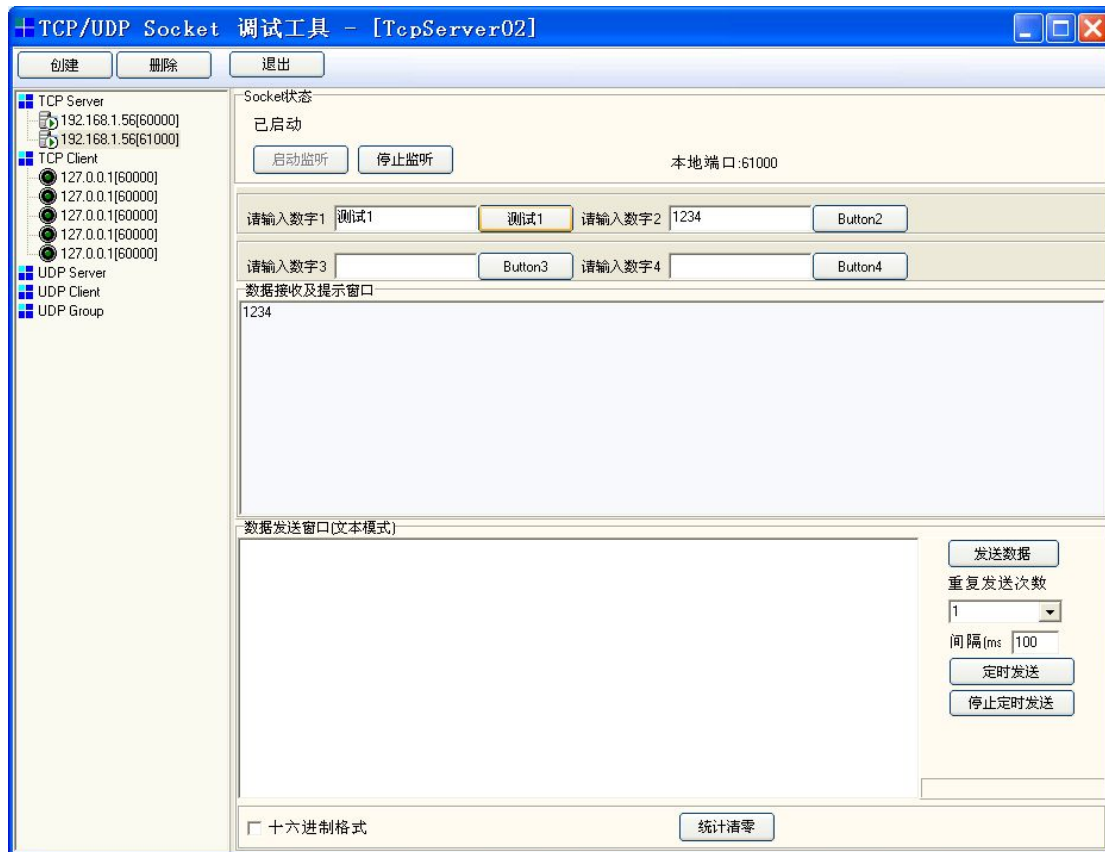
OnButtonClick 是按钮点击事件，任何一个按钮被点击时，都会触发这个事件。事件里面包含

了 socketid 和 buttonid;

举例：

```
function OnButtonClick(socketid,buttonid){  
  
    if (buttonid=='Button1'){  
  
        //将 Edit1 文本框的输入内容，设置为按钮 Button1 的标题  
  
        SetUIText(socketid,'Button1',GetUIText(socketid,'Edit1'));  
  
        //通过 socket 发送一串十六进制字符出去  
  
        Send(socketid, '0103000000001840A' );  
  
    }  
  
    if (buttonid=='Button2'){  
  
        //将 Edit2 文本框的内容，显示在数据接收和提示窗口中  
  
        Display(socketid,GetUIText(socketid,'Edit2'));  
  
    }  
  
}
```

在我们点击 Button1 和 Button2 按钮后，结果如下：



C) OnTimer 事件

Function OnTimerEvent (socketid , timername)

```
function OnTimerEvent(socketid,timername){
```

```
    if (socketid == 'TcpClient01'){
```

```
        Connect(socketid);
```

```
    }
```

```
}
```

例如，这个定时器就可以实现保持名为 TcpClient01 的这个 socket 一直连接到服务器端，如果断开，则自动重连。