

LAB 6-2 – EventHandler

I. Yêu cầu:

Sử dụng EventHandler hoàn thiện các chức năng cho chương trình gửi tin nhắn Server-Client bằng socket bất đồng bộ của Lab 6-1

Các yêu cầu:

- Hiện thị các thông tin trên Server form khi có Client kết nối, ngắt kết nối.
- Hiện thị tin nhắn của Client trên Server form.
- Xây dựng chương trình Client form.

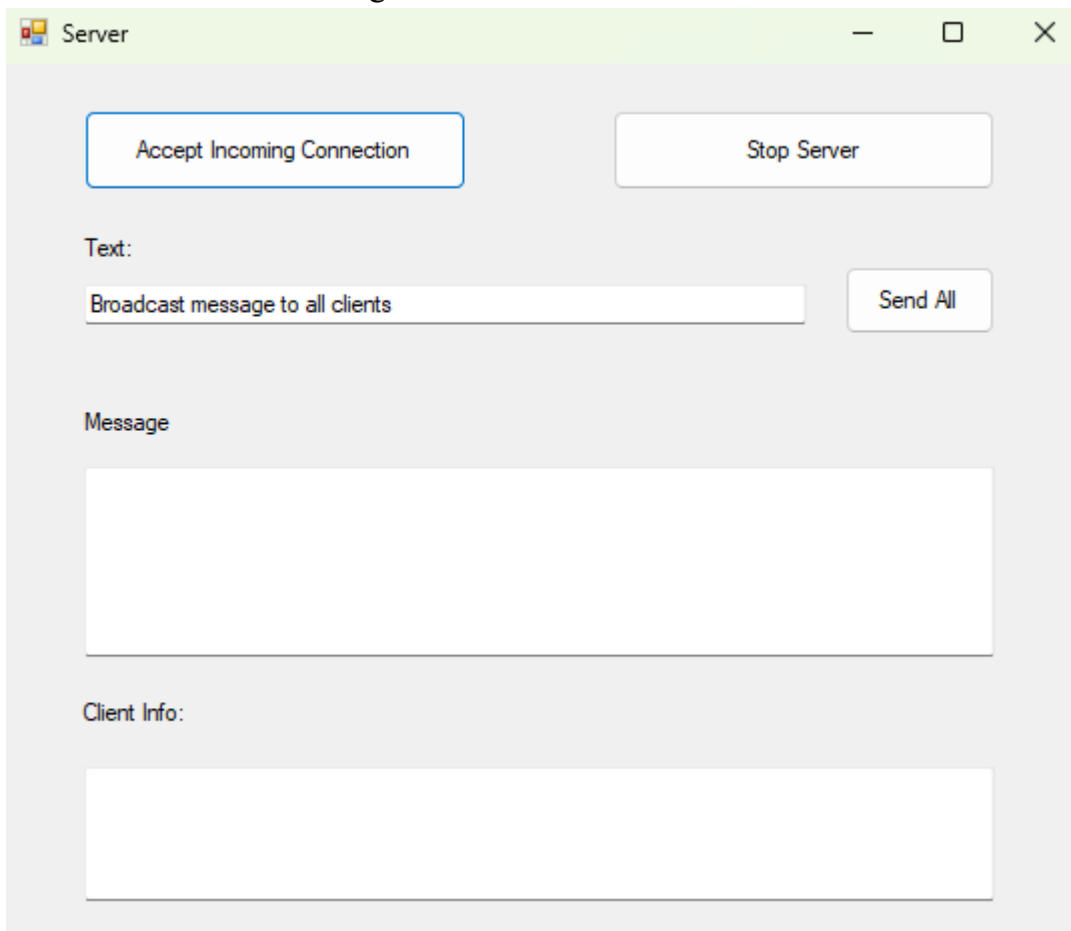
Tài liệu tham khảo: [Handle and raise events](#)

II. Bài tập:

1) Hiện thị thông tin trên Server form khi có Client kết nối thành công

a) Thiết kế bổ sung Server form:

- Thêm các control tương tự như sau:

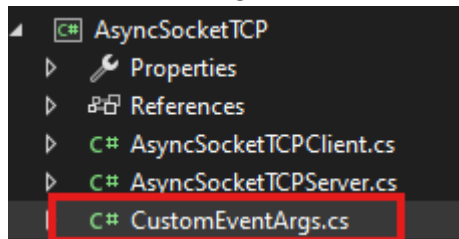


The screenshot shows a Windows form titled "Server". It contains the following controls:

- Two buttons at the top: "Accept Incoming Connection" and "Stop Server".
- A "Text:" label followed by a text box containing "Broadcast message to all clients" and a "Send All" button.
- A "Message" label followed by a large text area.
- A "Client Info:" label followed by a large text area.

- Đặt tên Textbox để chứa thông tin client (Client Info) là txtClientInfo
- b) Bổ sung lớp chứa dữ liệu của event

- Trong project AsyncSocketTCP thêm mới một class, đặt tên là CustomEventArgs.cs



- Thêm lớp ClientConnectedEventArgs chứa dữ liệu cho sự kiện khi có client kết nối thành công kèm phương thức khởi tạo cho lớp

```
public class ClientConnectedEventArgs : EventArgs
{
    2 references
    public string NewClient { get; set; }
    1 reference
    public ClientConnectedEventArgs(string _newClient)
    { _NewClient = _newClient; }
}
```

c) Khai báo sự kiện và trình xử lý sự kiện:

- Trong lớp AsyncSocketTCPServer, khai báo sự kiện đặt tên là ClientConnectedEvent

```
public EventHandler<ClientConnectedEventArgs> ClientConnectedEvent;
```

- Bổ sung phương thức kích hoạt sự kiện

```
protected virtual void OnClientConnectedEvent(ClientConnectedEventArgs e)
{
    EventHandler<ClientConnectedEventArgs> handler = ClientConnectedEvent;
    if (handler != null)
    {
        handler(this, e);
    }
}
```

- Kích hoạt sự kiện khi có client kết nối thành công: thêm đoạn mã sau vào phương thức *StartListeningForIncomingConnection*

```
var returnedByAccept = await mTCPListener.AcceptTcpClientAsync();
mClients.Add(returnedByAccept);

OnClientConnectedEvent(new ClientConnectedEventArgs(returnedByAccept.Client.RemoteEndPoint.ToString()));
```

d) Tại phía thức tiếp nhận sự kiện là Server form

- Thêm phương thức xử lý sự kiện

```
void HandleClientConnected(object sender, ClientConnectedEventArgs e)
{
    txtClientInfo.AppendText(string.Format("{0} - New client connected - {1}\r\n",
        DateTime.Now,
        e.NewClient));
}
```

- Thêm đoạn mã sau trong phương thức khởi tạo của form:

```
public Form1()
{
    InitializeComponent();
    mServer = new AsyncSocketTCPServer();
    mServer.ClientConnectedEvent += HandleClientConnected;
}
```

e) Thực thi chương trình để client kết nối và kiểm tra kết quả

The screenshot shows a Windows application window titled "Server". It contains several controls: a button labeled "Accept Incoming Connection", a button labeled "Stop Server", a text input field with the placeholder text "Broadcast message to all clients" and a "Send All" button next to it, a large empty text box labeled "Message", and a section labeled "Client Info:" containing a text box with the message "9/19/2024 10:43:29 PM - New client connected - 127.0.0.1:52325".

2) Thực hiện tương tự bổ sung các chức năng

- Khi có client ngắt kết nối, hiển thị thông báo và số Client còn lại
- Hiển thị tin nhắn của Client phía Server form (trong textbox Message)

3) Thiết kế và xây dựng chương trình Client Form tương tự như sau

The screenshot shows a Windows application window titled "Client". It contains several controls: input fields for "IP address" and "Port" with a "Connect" button next to them, a large empty text box labeled "Server", an input field labeled "Client" with a "Send" button next to it, and a "Send" button at the bottom right.