

LẬP TRÌNH MINICHAT

CHƯƠNG II, III

GIỚI THIỆU C.TRÌNH MINICHAT

- **Chức năng**
 - Cho phép nhiều user đăng ký vào các nhóm để trò chuyện với nhau.
- **Mô hình lựa chọn**
 - Client/server
- **Server**
 - Quản lý các nhóm và các user của từng nhóm.
 - phân phối chuỗi thông tin từ một user đến các user khác.
- **Client**
 - Giao tiếp với các user.
 - Cho phép họ đăng ký nhóm; gửi/nhận thông tin cho nhau.

GIỚI THIỆU C.TRÌNH MINICHAT

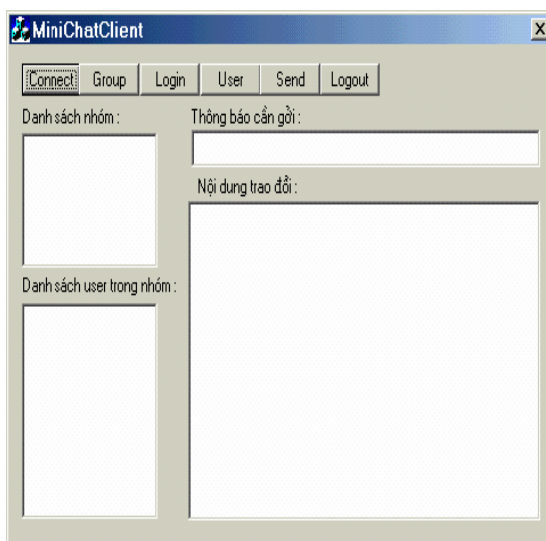
- **Giao thức dùng cho hệ thống MiniChat**

- Lệnh GLIST <CRLF>
- Lệnh ULIST <CRLF>
- Lệnh LOGIN <tên group>,<tên user> <CRLF>
- Lệnh SEND <string> <CRLF>
- Lệnh LOGOU <CRLF>

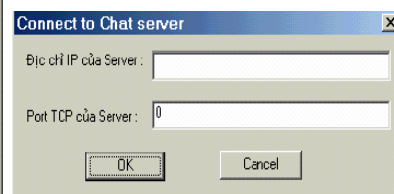
- **Dạng reply cho tất cả các request**

- N <chuỗi dữ liệu phụ kèm theo>
- N = 1: Thành công, N = 0: Thất bại

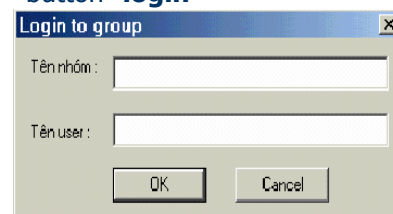
GIAO DIỆN C.TRÌNH MINICHAT



Cửa sổ nhập thông tin của button **"Connect"**

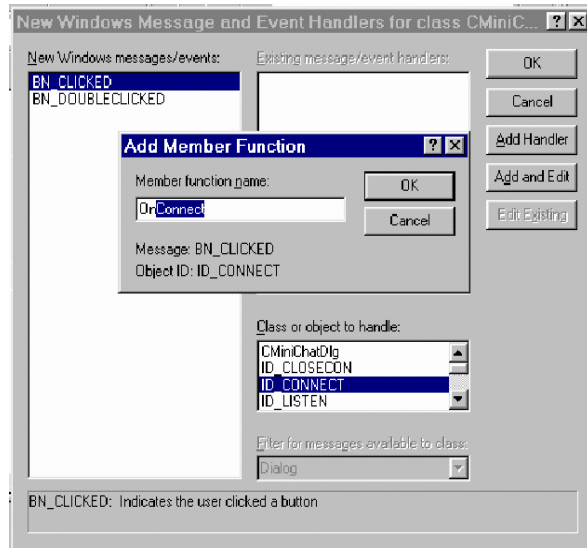


Cửa sổ nhập thông tin của button **"login"**



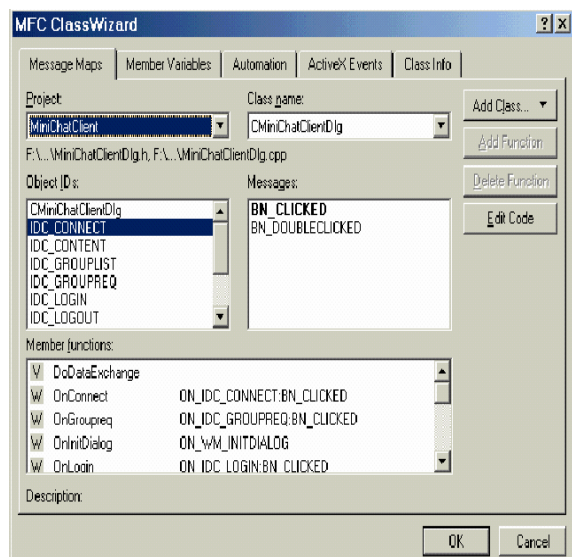
XỬ LÝ BIẾN CỐ TRÊN VC++

- Tạo hàm xử lý biến cố cho từng button bằng cách chọn từng button, chọn mục event trong cửa sổ Properties, cửa sổ sau xuất hiện:



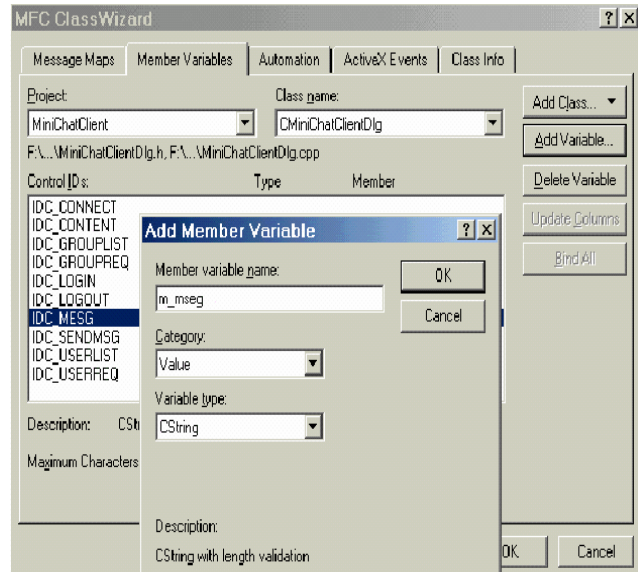
XỬ LÝ BIẾN CỐ TRÊN VC++ (t.t)

- Phương pháp chính quy để khai báo biến và hàm xử lý biến cố với các phần tử giao diện là dùng menu View>ClassWizard, cửa sổ ClassWizard xuất hiện, trang MessageMap cho phép khai báo các hàm xử lý biến cố:



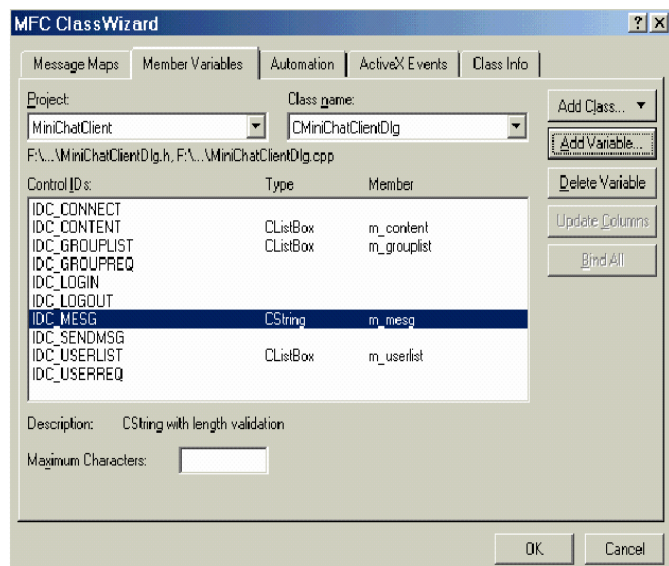
XỬ LÝ BIẾN CỐ TRÊN VC++ (t.t)

- Để tạo các biến dữ liệu kết hợp với các control, chọn project, class chứa biến, trang Member variables, sau đó chọn từng ID phần tử rồi ấn nút button “Add variable”, cửa sổ sau xuất hiện:



XỬ LÝ BIẾN CỐ TRÊN VC++ (t.t)

- Kết quả tạo 4 biến kết hợp với 4 phần tử giao diện.



MINICHAT SERVER TRÊN VC++

```
BOOL CMiniChatServerDlg::OnInitDialog() {  
    ...  
    // Tao socket moi, neu that bai bao sai  
    ser_sock=socket(AF_INET,SOCK_STREAM,0);  
    if(ser_sock==INVALID_SOCKET) {  
        MessageBox("Khong tao duoc socket");  
        return TRUE;  
    }  
  
    // Thiet lap dia chi diem dau mut va bind no voi socket  
    SOCKADDR_IN local_addr;  
    local_addr.sin_family=AF_INET;  
    local_addr.sin_port=256;  
    local_addr.sin_addr.s_addr=INADDR_ANY;  
    if(bind(ser_sock,(LPSOCKADDR)&local_addr,sizeof(local_addr))==SOCKET_ERROR) {  
        MessageBox("Khong bind socket duoc");  
        return TRUE;  
    }  
}
```

MINICHAT SERVER TRÊN VC++ (t.t)

```
// Khai bao so yeu cau ket noi dong thoi  
if(listen(ser_sock,10)==SOCKET_ERROR) {  
    MessageBox("Khong listen duoc");  
    return TRUE;  
}  
  
// Khai bao nhan du lieu bat dong bo + dong cau noi bat dong bo  
if (WSAAsyncSelect(ser_sock, m_hWnd, WSA_ACCEPT, FD_ACCEPT) > 0) {  
    MessageBox("Error on WSAAsyncSelect()");  
    closesocket(ser_sock);  
}  
...  
}
```

MINICHAT SERVER TRÊN VC++ (t.t)

```
// Accept 1 yêu cầu nối kết
void CMiniChatServerDlg::OnAccept(void) {
    SOCKADDR_IN remote_addr;
    SOCKET sock;
    // Cho kết nối
    int len=sizeof(remote_addr);
    sock=accept(ser_sock,(LPSOCKADDR)&remote_addr,&len);
    if(sock==INVALID_SOCKET) {
        MessageBox("Không accept được");
        return;
    }
    T_UserRec *puser = new(T_UserRec);
    puser->sock = sock;
    puser->next = sock_no_user;
    sock_no_user = puser;
    // Khởi tạo nhận dữ liệu bất đồng bộ + đồng bộ câu nói bất đồng bộ
    if (WSAAsyncSelect(sock, m_hWnd, WSA_RDCLOSE, FD_READ|FD_CLOSE) > 0) {
        MessageBox("Error on WSAAsyncSelect()");
        closesocket(sock);
    }
}
```

MINICHAT SERVER TRÊN VC++ (t.t)

```
// Đọc vào request và xử lý
void CMiniChatServerDlg::Request_Process(SOCKET sock) {
    int status;
    char mesg[MSG_LENGTH];
    status = recv(sock, mesg, MSG_LENGTH, 0);
    if (status==0) return;
    mesg[status] = 0;
    if (strncmp(mesg,"LOGIN",5)==0) { // login
        Do_login(sock,mesg);
    } else if (strncmp(mesg,"LOGOU",5)==0) { // logout
        Do_logout(sock);
    } else if (strncmp(mesg,"GLIST",5)==0) { // group list
        Do_glist(sock);
    } else if (strncmp(mesg,"ULIST",5)==0) { // user list
        Do_ulist(sock);
    } else { // broadcast message
        Do_broadcastMesg(sock,mesg);
    }
}
```

KỸ THUẬT JAVA Multithread

● Thread

- Một luồng thực thi trong một chương trình.
- Máy ảo JVM cho phép một ứng dụng có nhiều luồng thực thi đồng thời.

● Có 2 cách dùng Java multithread (đa luồng):

- Khai báo một lớp kế thừa từ lớp Thread và override method Thread.run().
- Khai báo một lớp hiện thực interface Runnable và method Runnable.run()

KỸ THUẬT JAVA Multithread

```
1. class PrimeThread extends Thread {  
2.     long minPrime;  
3.     PrimeThread( long minPrime ) {  
4.         this.minPrime = minPrime;  
5.     }  
6.     public void run( ) {  
7.         // compute primes larger than minPrime  
8.         ...  
9.     }  
10. }  
  
11. PrimeThread p = new PrimeThread(143);  
12. p.start();
```

KỸ THUẬT JAVA Multithread

```

1. class PrimeRun implements Runnable {
2.     long minPrime;
3.     PrimeRun ( long minPrime ) {
4.         this.minPrime = minPrime;
5.     }

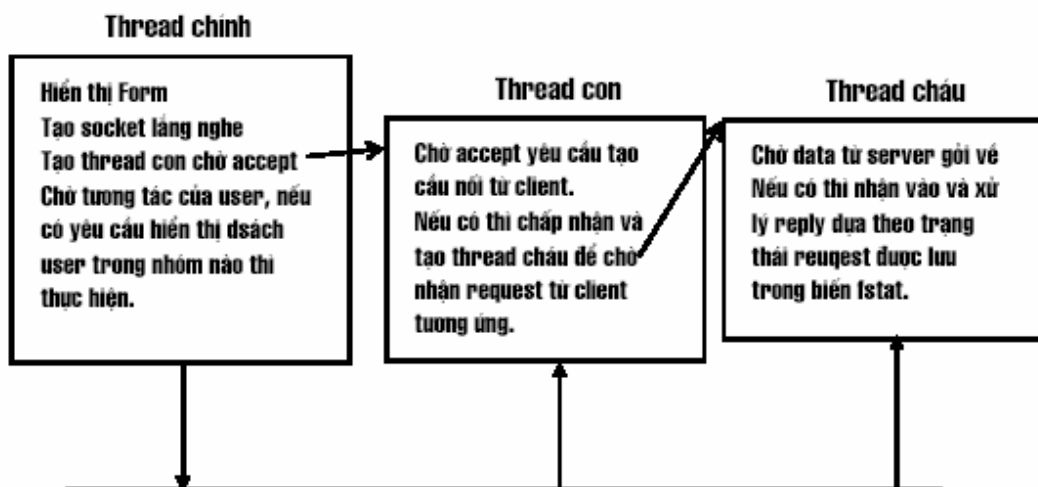
6.     public void run() {
7.         // compute primes larger than minPrime
8.         ...
9.     }
10. }

11. PrimeRun p = new PrimeRun(143);
12. new Thread(p).start();

```

MINICHAT SERVER TRÊN JBuilder

Mô hình các thread ở Server :



MINICHAT SERVER TRÊN JBuilder

```
//Constructor của frame
public MiniChatServerDlg() {
    ...
    // Tao sersocket lang nghe cho server
    try {
        serverSocket = new ServerSocket( SERVER_PORT, 100 );
        DefaultListModel lmContent = (DefaultListModel)jlbContent.getModel();
        lmContent.addElement("Server listening on port " + SERVER_PORT + "
        ...");
        // tạo thread con để chờ
        new ServerAcceptThread(this,serverSocket).start();
    } // end try
    // handle exception creating server and connecting clients
    catch ( IOException ioException ) {
        ioException.printStackTrace();
    }
    ...
}
```

MINICHAT SERVER TRÊN JBuilder (t.t)

```
public class ServerAcceptThread extends Thread {
    ServerSocket serverSocket;
    MiniChatServerDlg serverChat;
    public ServerAcceptThread(MiniChatServerDlg server, ServerSocket sock) {
        serverSocket = sock;
        serverChat = server;
    }
    public void run() {
        T_UserRec puser;
        try {
            // listen for clients constantly
            while (true) {
                // accept new client connection
                Socket clientSocket = serverSocket.accept();
                puser = new T_UserRec();
                puser.sock = clientSocket;
                puser.next = serverChat.m_sock_no_user;
                serverChat.m_sock_no_user = puser;
            }
        }
    }
}
```

MINICHAT SERVER TRÊN JBuilder (t.t)

```
// create new ReceivingThread for receiving messages from client
new ReceivingThread(serverChat, clientSocket).start();
// print connection information
DefaultListModel lmContent =
(DefaultListModel)serverChat.jlbContent.getModel();
lmContent.addElement("Connection received from: " +

        clientSocket.getInetAddress());
serverChat.SendMessage(clientSocket,"Request accepted");
} // end while
}
// handle exception creating server and connecting clients
catch ( IOException ioException ) {
    ioException.printStackTrace();
}
}
}
```

TỔNG KẾT

- Giới thiệu chương trình MiniChat
- Cách viết chương trình trên Visual C++
- Cách viết chương trình trên JBuilder