**WCF - 채팅프로그램**

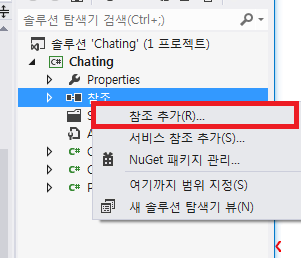
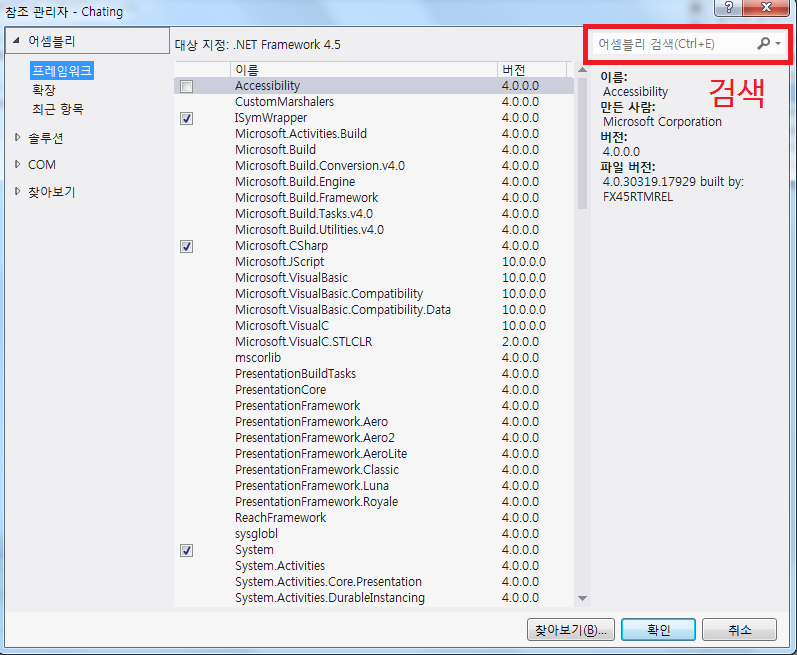
**< 1:다 채팅 서버> (.Net Framework 4.5버전 이상 요구)**

**\*\*비주얼 스튜디오를 실행 시 반드시 관리자 권한으로 실행시켜 주세요!**

**프로젝트 생성 : 비주얼 스튜디오 – 콘솔 응용프로그램**

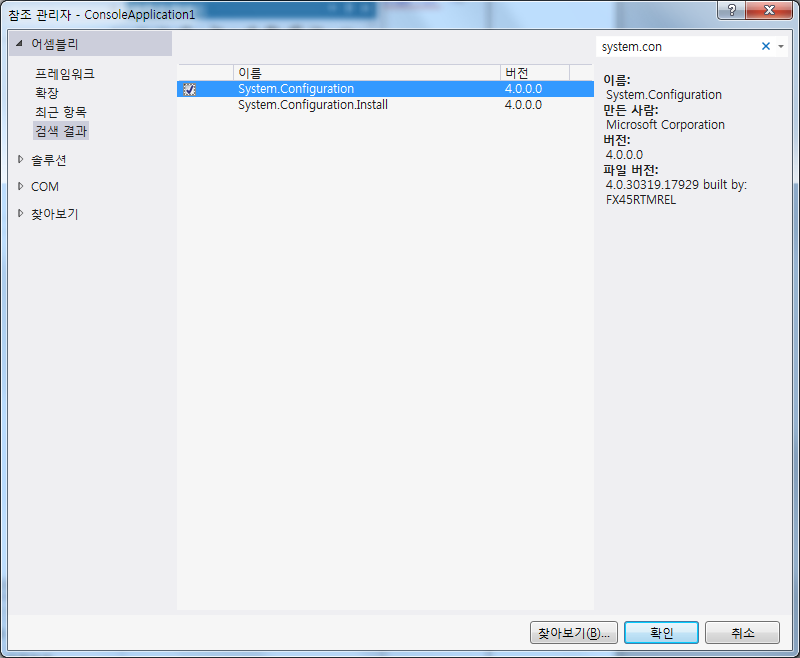
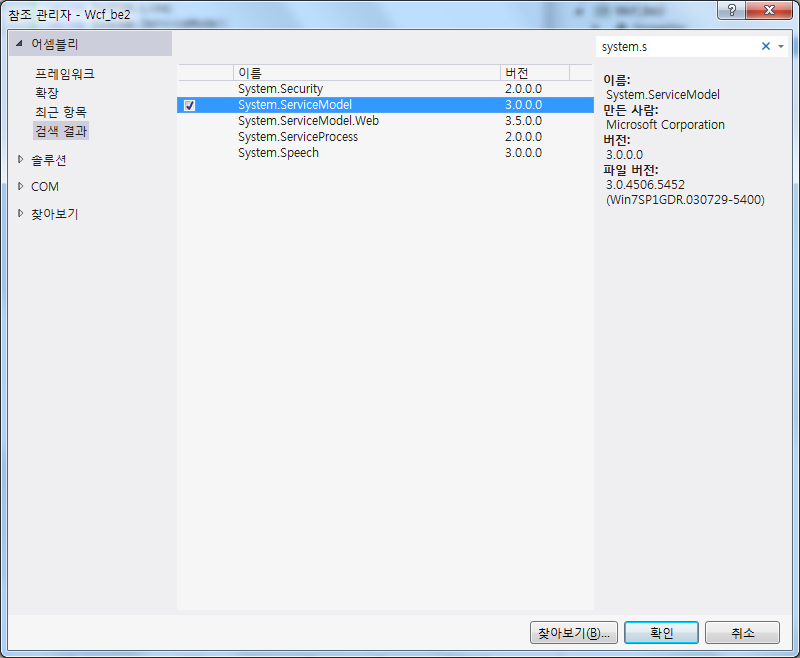
**프로젝트 생성 후 참조추가 필수**

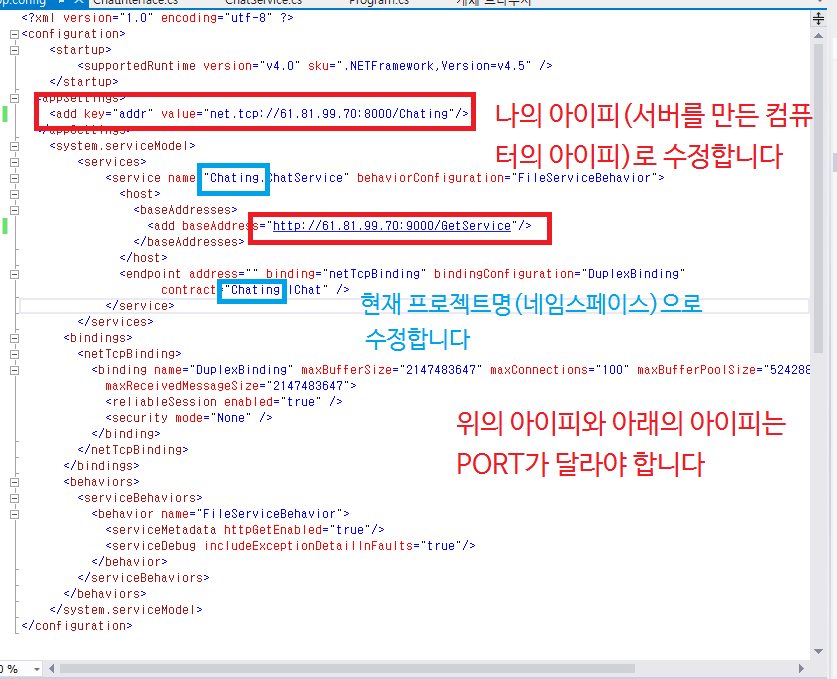
**비주얼 스튜디오의 인터페이스 설정에 따라 배경색이 다를 수 있습니다**

** **

**솔루션 탐색기에서 ‘참조’를 우클릭 하면 참조 추가 가능하며,**

**참조 관리자의 우측 상단에서 참조검색을 통해 쉽게 찾을 수 있습니다**

**System.ServiceModel System.Configration**

****

**App.config**

|  |
| --- |
| **APP.config 파일** |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <startup>  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />  </startup>  <appSettings>  <add key="addr" value="net.tcp://61.81.99.75:8000/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="Chating.ChatService" behaviorConfiguration="FileServiceBehavior">  <host>  <baseAddresses>  <add baseAddress="http://61.81.99.75:9000/GetService"/>  </baseAddresses>  </host>  <endpoint address="" binding="netTcpBinding" bindingConfiguration="DuplexBinding"  contract="Chating.IChat" />  </service>  </services>  <bindings>  <netTcpBinding>  <binding name="DuplexBinding" maxBufferSize="2147483647" maxConnections="100" maxBufferPoolSize="524288"  maxReceivedMessageSize="2147483647">  <reliableSession enabled="true" />  <security mode="None" />  </binding>  </netTcpBinding>  </bindings>  <behaviors>  <serviceBehaviors>  <behavior name="FileServiceBehavior">  <serviceMetadata httpGetEnabled="true"/>  <serviceDebug includeExceptionDetailInFaults="true"/>  </behavior>  </serviceBehaviors>  </behaviors>  </system.serviceModel>  </configuration> |

|  |
| --- |
| **메인 cs 파일** |
| using System;  using System.Configuration;  using System.ServiceModel;  namespace Chating  {  class Program  {  static void Main(string[] args)  {  //Address  Uri uri = new Uri(ConfigurationManager.AppSettings["addr"]);  //Contract-> Setting  //Binding -> App.Config  ServiceHost host = new ServiceHost(typeof(Chating.ChatService), uri);  //오픈  host.Open();  Console.WriteLine("채팅 서비스를 시작합니다. {0}", uri.ToString());  Console.WriteLine("http://61.81.99.75:9000/GetService");  Console.WriteLine("멈추시려면 엔터를 눌러주세요..");  Console.ReadLine();  //서비스  host.Abort();  host.Close();  }  }  } |

**Class 명 : ChatService**

Class 파일을 새로 생성하세요

|  |
| --- |
| **서브 cs 파일 - 1** |
| using System;  using System.ServiceModel;  using System.Collections;  namespace Chating  {  class ChatService : IChat  {  //델리게이트 선언  public delegate void Chat(int idx, string msg, string type);  //동기화 작업을 위해서 가상의 객체 생성  private static Object syncObj = new Object();  //채팅방에 있는 유저 이름 목록  private static ArrayList Chatter = new ArrayList();  //델리게이트 =========================================================  // 개인용 델리게이트  private Chat MyChat;  //전체에게 보낼 정보를 담고 있는 델리게이트  private static Chat List;  IChatCallback callback = null; //  //========================== IChat 메서드 =======================================================  #region 1. Join(로그인하기)  public bool Join(int idx)  {  MyChat = new Chat(UserHandler);  lock (syncObj)  {  if (!Chatter.Contains(idx)) //이름이 기존 채터에 있는지 검색한다.  {  //2. 사용자에게 보내 줄 채널을 설정한다.  callback = OperationContext.Current.GetCallbackChannel<IChatCallback>();  //현재 접속자 정보를 모두에게 전달  BroadcastMessage(idx, "", "UserEnter");  //델리게이터 추가(향후 데이터 수신이 가능하도록 구성)  List += MyChat;    return true;  }  return false;  }  }  #endregion  #region 2. Say(메시지 보내기)  public void Say(int idx, string msg)  {  BroadcastMessage(idx, msg, "Receive");  }  #endregion  #region 3. Leave(로그아웃 하기)  public void Leave(int idx)  {      //메시지 수신에서 제외  List -= MyChat;  //모든 사람에게 전송  string msg = string.Format(idx + "이가 나갔습니다");  BroadcastMessage(idx, msg, "UserLeave");  }  #endregion  //===================================================================================================    private void BroadcastMessage(int idx, string msg, string msgType)  {  if (List != null)  {  //현재 이벤트들을 전달한다.  foreach (Chat handler in List.GetInvocationList())  {  handler.BeginInvoke(idx, msg, msgType, new AsyncCallback(EndAsync), null);  }  }  }  private void UserHandler(int idx, string msg, string msgType)  {  try  {  //클라이언트에게 보내기  switch (msgType)  {  case "Receive":  callback.Receive(idx, msg);  break;  case "UserEnter":  callback.UserEnter(idx);  break;  case "UserLeave":  callback.UserLeave(idx);  break;  }  }  catch//에러가 발생했을 경우  {  Leave( idx);  }  }  private void EndAsync(IAsyncResult ar)  {  Chat d = null;  try  {  System.Runtime.Remoting.Messaging.AsyncResult asres = (System.Runtime.Remoting.Messaging.AsyncResult)ar;  d = ((Chat)asres.AsyncDelegate);  d.EndInvoke(ar);  }  catch  {  List -= d;  }  }  }  } |

**Class 명 : IChat**

Class 파일을 새로 생성하세요

|  |
| --- |
| **서브 cs 파일 - 2** |
| using System.ServiceModel;  namespace Chating  {  #region 1. 메세지 관련 Contract InterFace (클라이언트->서버)  [ServiceContract(SessionMode = SessionMode.Required, CallbackContract = typeof(IChatCallback))]  public interface IChat  {  [OperationContract(IsOneWay = false, IsInitiating = true, IsTerminating = false)]  bool Join(int idx);  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = false)]  void Say(int idx, string msg);  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = true)]  void Leave(int idx);  }  #endregion  #region 2. 클라이언트에 콜백할 CallBackContract (서버->클라이언트)  public interface IChatCallback  {  [OperationContract(IsOneWay = true)]  void Receive(int idx, string message);  [OperationContract(IsOneWay = true)]  void UserEnter(int idx);    [OperationContract(IsOneWay = true)]  void UserLeave(int idx);  }  #endregion  } |

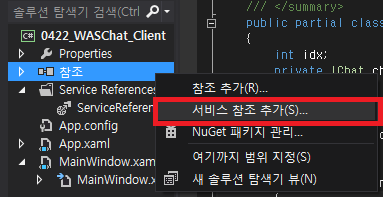
**\*프로그램 실행 시 방화벽 허용 메시지가 뜰 경우 반드시 ‘허용’을 클릭하여 서버를 열어줘야 합니다**

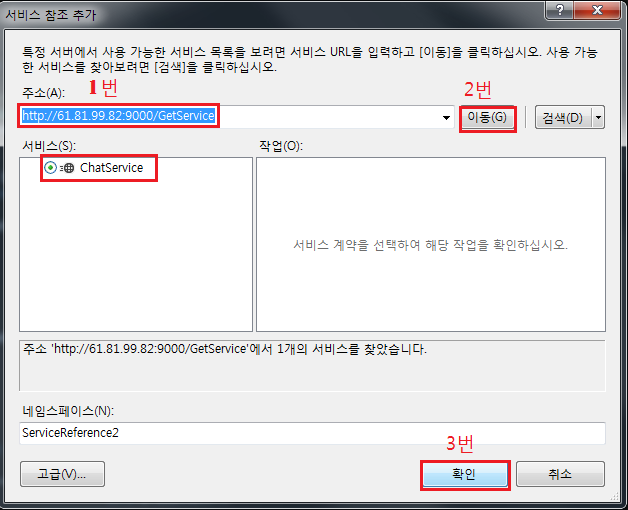
**만약 취소를 누를 시 다시 물어보지 않기 때문에 프로젝트를 재생성 해야 합니다**

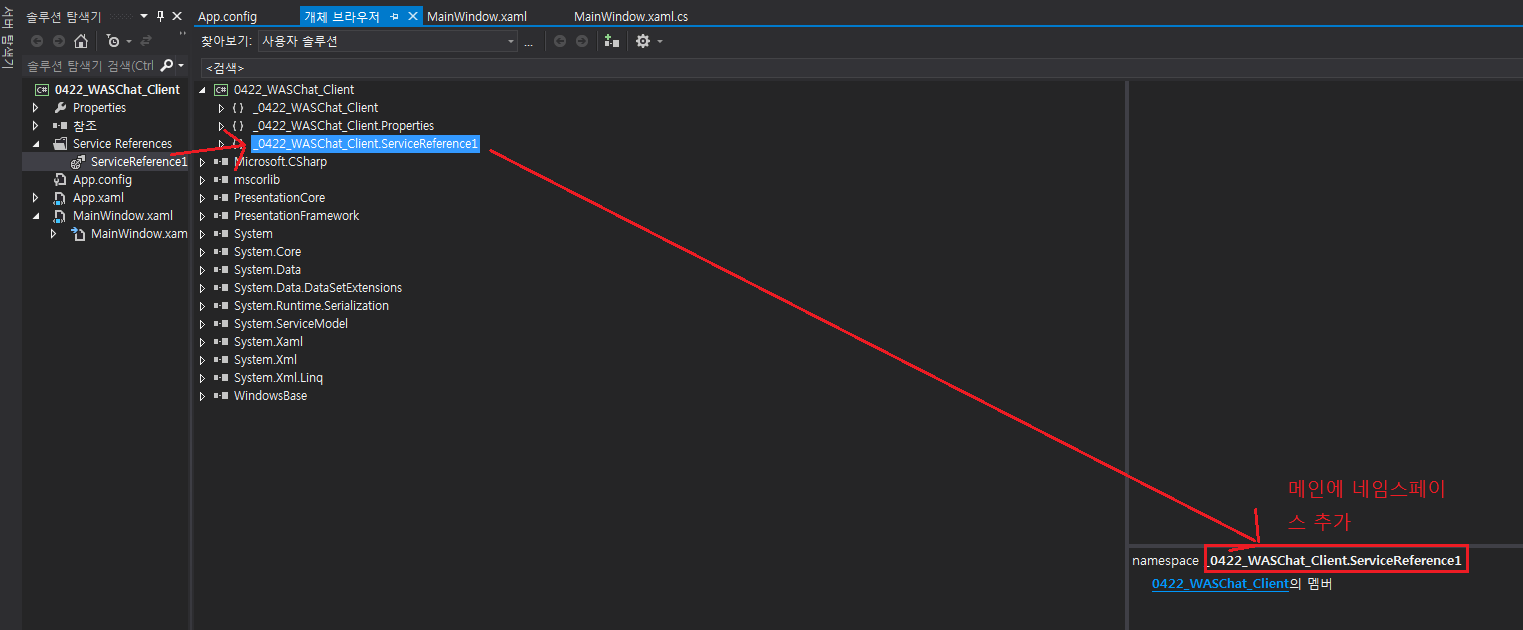
**< 1:다 채팅 클라이언트>**

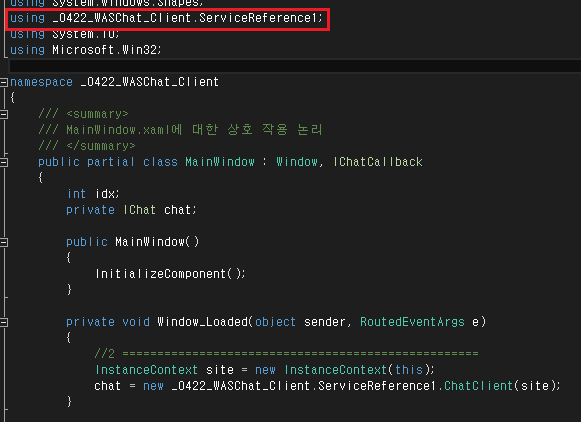
**프로젝트 생성 : 비주얼 스튜디오 – WCF 응용 프로그램 서버 프로젝트와 다르니까 주의 하세요**

**서비스 참조추가 필수!**

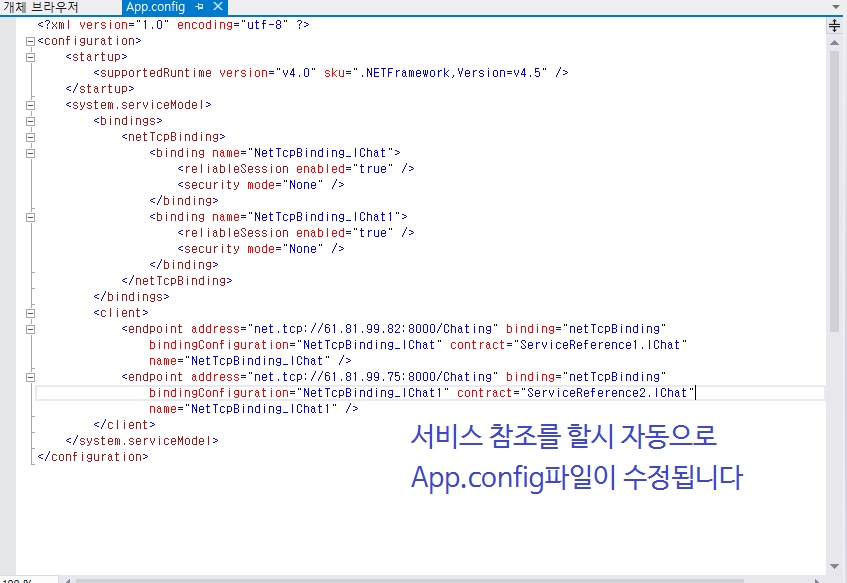


****

****

****

**\*서비스 참조 추가를 할 시에 서버가 열려 있어야 합니다**

****

**App.config**

|  |
| --- |
| **메인 xaml 파일** |
| <Window x:Class="\_0422\_WASChat\_Client.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="MainWindow" Height="530" Width="381" Loaded="Window\_Loaded">  <Grid>  <TextBox Name="seatbox" HorizontalAlignment="Left" Height="22" Margin="86,74,0,0" TextWrapping="Wrap" Text="좌석을 선택하세요." VerticalAlignment="Top" Width="158" TextAlignment="Center"/>  <Button Name="btnJoin" Content="로그인" HorizontalAlignment="Left" Margin="250,46,0,0" VerticalAlignment="Top" Width="74" Height="50" Click="btnJoin\_Click"/>  <ListBox Name="chatlist" HorizontalAlignment="Left" Height="272" Margin="62,130,0,0" VerticalAlignment="Top" Width="262"/>  <TextBox Name="msgbox" HorizontalAlignment="Left" Height="24" Margin="62,406,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="198"/>  <Button Name="btnSend" Content="전송" HorizontalAlignment="Left" Margin="264,406,0,0" VerticalAlignment="Top" Width="60" Click="btnSend\_Click"/>  </Grid>  </Window> |
| **메인 cs파일** |
| using System;  using System.ServiceModel;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Input;  using \_0422\_WASChat\_Client.ServiceReference2;  namespace \_0422\_WASChat\_Client  {  // MainWindow.xaml에 대한 상호 작용 논리  public partial class MainWindow : Window, IChatCallback  {  int idx;    private IChat chat;  public MainWindow()  {  InitializeComponent();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  //2 ===================================================  InstanceContext site = new InstanceContext(this);  chat = new \_0422\_WASChat\_Client.ServiceReference2.ChatClient(site);  btnSend.IsEnabled = false;  }  #region IChatCallback 인터페이스 함수 생성  public void Receive(int idx, string message)  {  string msgtemp = string.Format("{0}", message);  chatlist.Items.Add(msgtemp);  }  public void UserEnter(int idx)  {  string msgtemp = string.Format("{0}님이 로그인하셨습니다.", idx);  chatlist.Items.Add(msgtemp);  }  public void UserLeave(int idx)  {  string msgtemp = string.Format("{0}님이 로그아웃하셨습니다.",idx);  chatlist.Items.Add(msgtemp);  }  #endregion  #region 로그인/로그아웃 핸들러  private void btnJoin\_Click(object sender, RoutedEventArgs e)  {  if ((string)btnJoin.Content == "로그인")  {  this.Connect();  string msgtemp = string.Format("{0}님이 로그인하셨습니다.", idx);  btnSend.IsEnabled = true;  }  else this.DisConnect();  }  private void Connect()  {  try  {  idx = int.Parse(seatbox.Text);  //서버 접속  bool data = chat.Join(idx);    btnJoin.Content = "로그아웃";  string login = string.Format("{0}님이 로그인하셨습니다.", seatbox.Text);  }  catch(Exception ex)  {  MessageBox.Show("접속 오류 :{0}", ex.Message);  }  }  private void DisConnect()  {  try  {  chat.Leave(idx);  btnJoin.Content = "로그인";  string logout = string.Format("{0}님이 로그아웃하셨습니다.", seatbox.Text);  chatlist.Items.Add(logout);  btnSend.IsEnabled = false;  }  catch (Exception ex)  {  MessageBox.Show("나가기 오류 :{0}", ex.Message);  }  }  #endregion  #region 마우스 이벤트 핸들러  private void Label\_MouseDown(object sender, MouseButtonEventArgs e)  {  seatbox.Text = (string)((Label)sender).Content;  }  // 메시지 전송  private void btnSend\_Click(object sender, RoutedEventArgs e)  {  string msg = msgbox.Text;  string temp = string.Format("[{0}]", msg);  // chatlist.Items.Add(temp);  chat.Say(idx, msg);  msgbox.Clear();  }    #endregion  }  } |