|  |
| --- |
| **WCF - 채팅프로그램**  < 1:1 채팅 서버> (.Net Framework 4.5버전 이상 요구) |

1. **프로젝트 구성**

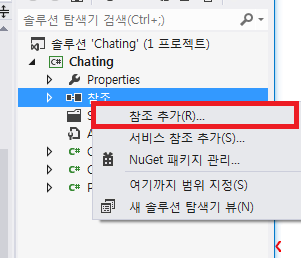
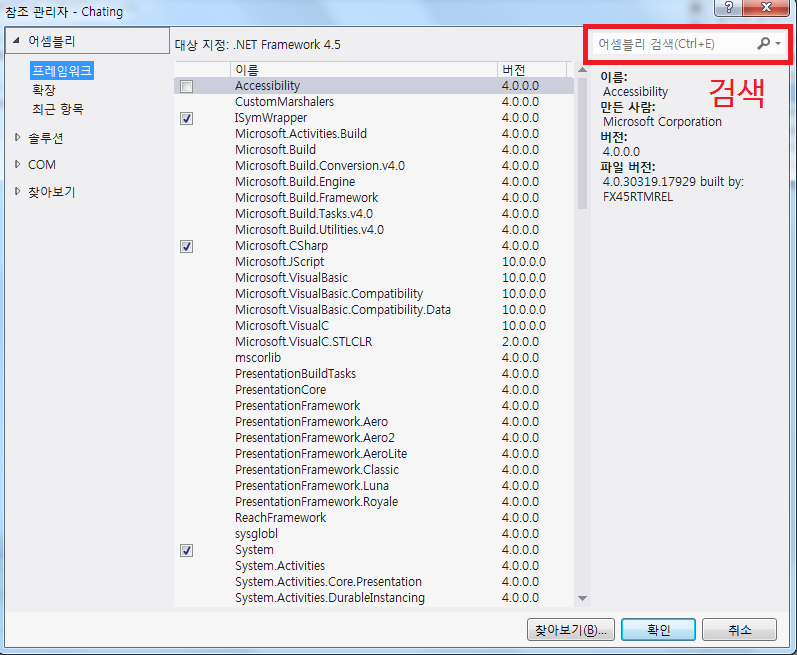
- 비주얼 스튜디오를 실행 시 반드시 관리자 권한으로 실행

- 프로젝트 생성 : 비주얼 스튜디오 – 콘솔 응용프로그램

- 프로젝트 생성 후 어셈블리 참조 추가(2개)

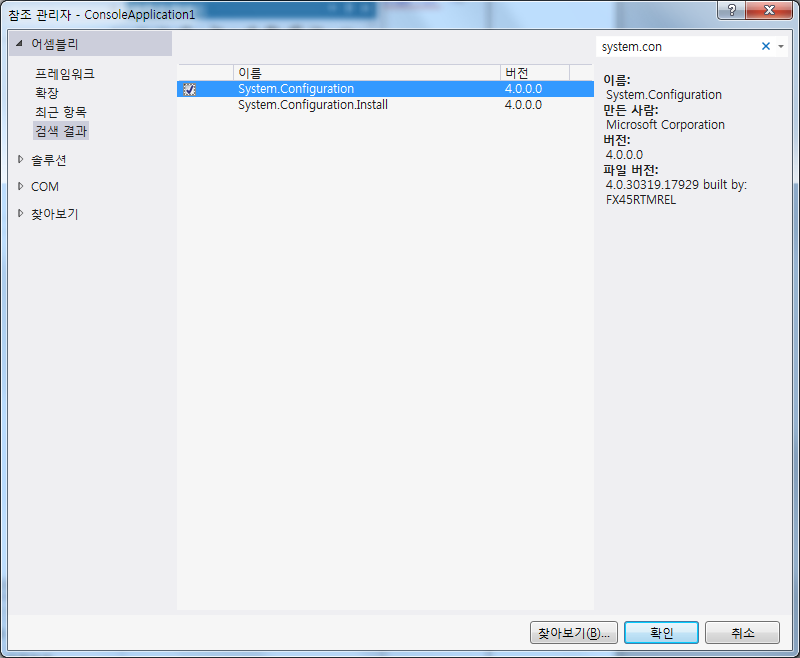
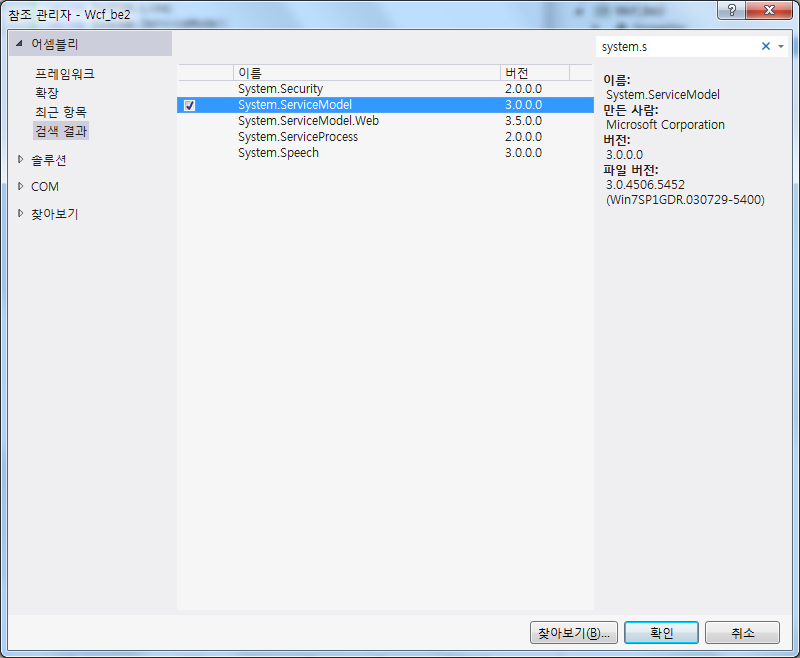
System.ServiceModel

System.Configration

** **

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

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

**System.ServiceModel System.Configration**

1. **인터페이스 정의(이중 계약)**

인터페이스 생성 : IChat, IChatCallback

|  |
| --- |
| **IChat.cs [파일 추가]** |
| using System;  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);  }  #endregion  } |

SessionMode = SessionMode.Required : 세션모드를 사용하겠다.

IsInitiating : 반드시 먼저 호출되어야 할 메서드(세션을 시작됨)

IsTerminating : 반드시 마지막으로 호출되어야 할 메서드(세션을 종료함)

1. **1대 1 통신 클래스 구현**

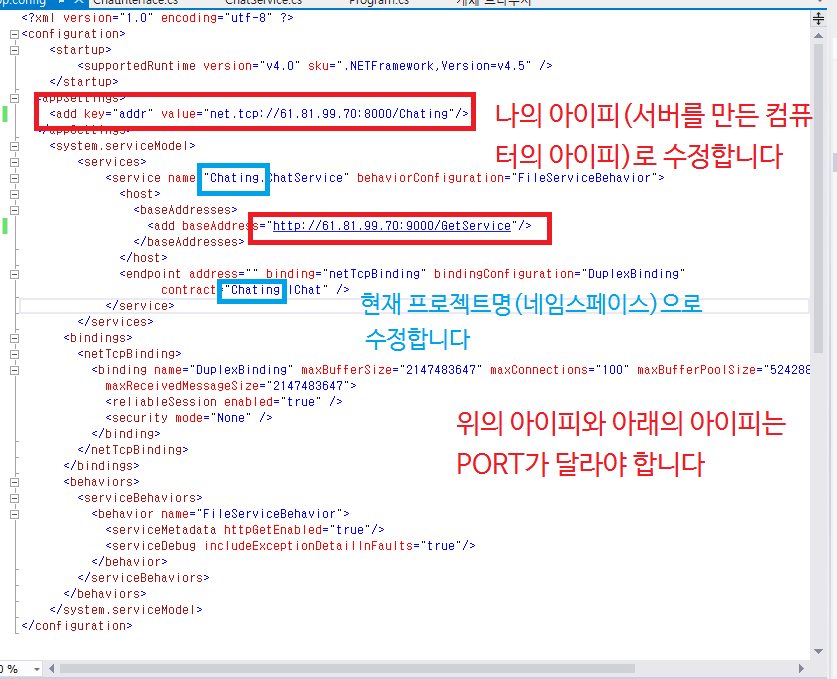
|  |
| --- |
| **WbChat.cs [파일 추가]** |
| using System;  namespace \_0427\_WCFChat  {  //델리게이트 선언  public delegate void Chat(string nickname, string msg, string type);  internal class WbChat  {  // 개인용 델리게이트  private Chat MyChat = null;  IChatCallback callback = null;  public bool Join(IChatCallback call, string nickname) {  MyChat = new Chat(UserHandler);  callback = call;  return true;  }  public void Leave(string nickname) {  Chat d = null;  MyChat -= d;  }    public void BroadcastMessage(string nickname, string msg, string msgType) {  MyChat.BeginInvoke(nickname, msg, msgType, new AsyncCallback(EndAsync), null);  }  //BroadcastMessage()에서 호출 : Client 메서드 호출  private void UserHandler(string nickname, string msg, string msgType) {  try  {  //클라이언트에게 보내기  switch (msgType)  {  case "Receive": callback.Receive(nickname, msg); break;  case "UserEnter": callback.UserEnter(nickname); break;  }  }  catch {  Console.WriteLine("{0} 에러", nickname);  }  }  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  {  }  }  }  } |

1. **서비스 객체 구현(이중 계약)**

서비스 객체 생성 : ChatService

|  |
| --- |
| **ChatService.cs [파일추가]** |
| using System;  using System.ServiceModel;  namespace \_0427\_WCFChat  {  internal class ChatService : IChat  {  private IChatCallback callback = null;  private WbChat wbchat = new WbChat();  #region IChat  public bool Join(string nickname)  {  //사용자에게 보내 줄 채널을 설정한다.  callback = OperationContext.Current.GetCallbackChannel<IChatCallback>();  if (wbchat.Join(callback, nickname) == false)  return false;    //현재 접속자 클라이언트에게 전달  wbchat.BroadcastMessage(nickname, "", "UserEnter");  return true;  }  public void Leave(string nickname)  {  wbchat.Leave(nickname);  }  public void Say(string nickname, string msg)  {  wbchat.BroadcastMessage(nickname, msg, "Receive");  }  #endregion  }  } |

1. **Config파일 구성**

****

|  |
| --- |
| **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://192.168.0.9:8000/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="Chating.ChatService" behaviorConfiguration="FileServiceBehavior">  <host>  <baseAddresses>  <add baseAddress=" http://192.168.0.9/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> |

1. **Hosting(Self Hosting)**

|  |
| --- |
| **메인 cs 파일** |
| using System;  using System.Configuration;  using System.ServiceModel;  namespace \_0427\_WCFChat  {  internal class Program  {  static void Main(string[] args)  {  Uri uri = new Uri(ConfigurationManager.AppSettings["addr"]);  ServiceHost host = new ServiceHost(typeof(\_0427\_WCFChat.ChatService), uri);  //오픈  host.Open();  Console.WriteLine("채팅 서비스를 시작합니다. {0}", uri.ToString());  Console.WriteLine("http://192.168.0.9/GetService ");  Console.WriteLine("멈추시려면 엔터를 눌러주세요..");  Console.ReadLine();  //서비스  host.Abort();  host.Close();  }  }  } |

\*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

|  |
| --- |
| **채팅프로그램- Client**  **< 1:1 채팅 클라이언트>** |

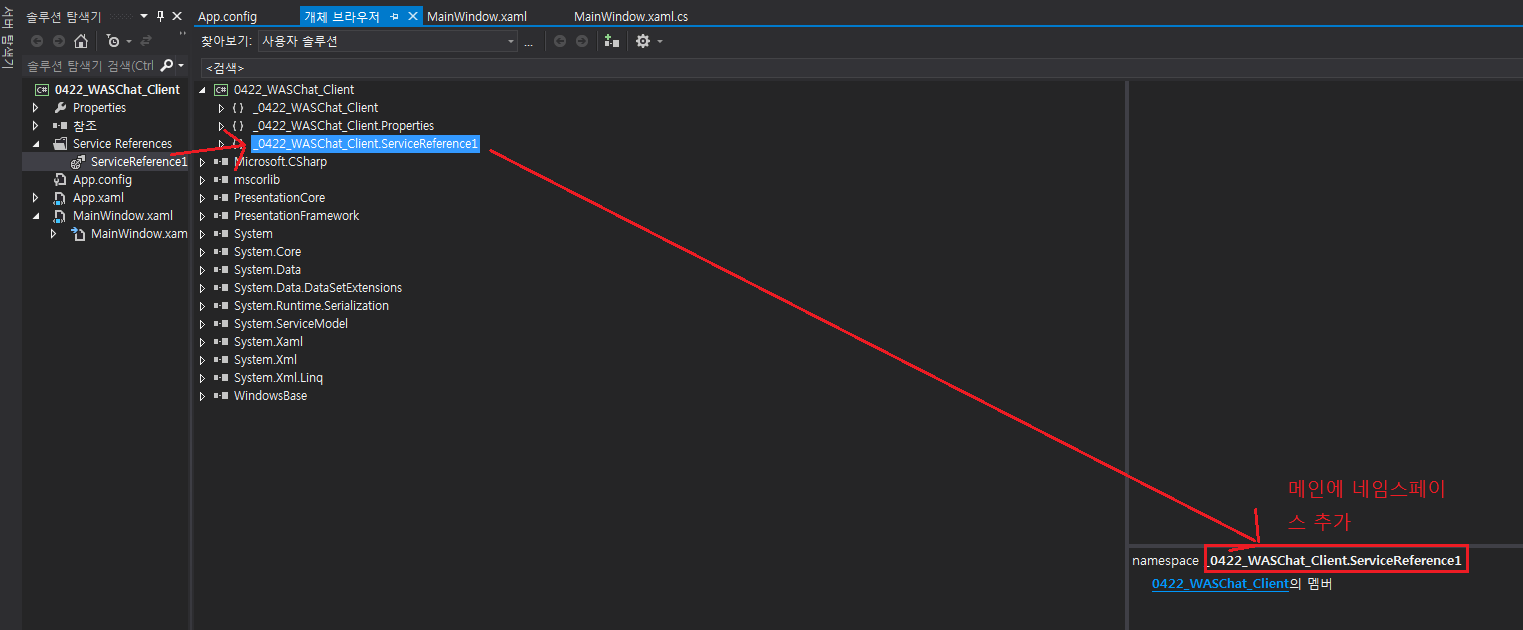
1. **프로젝트 구성**

- 프로젝트 생성 : 비주얼 스튜디오 – WPF 응용 프로그램

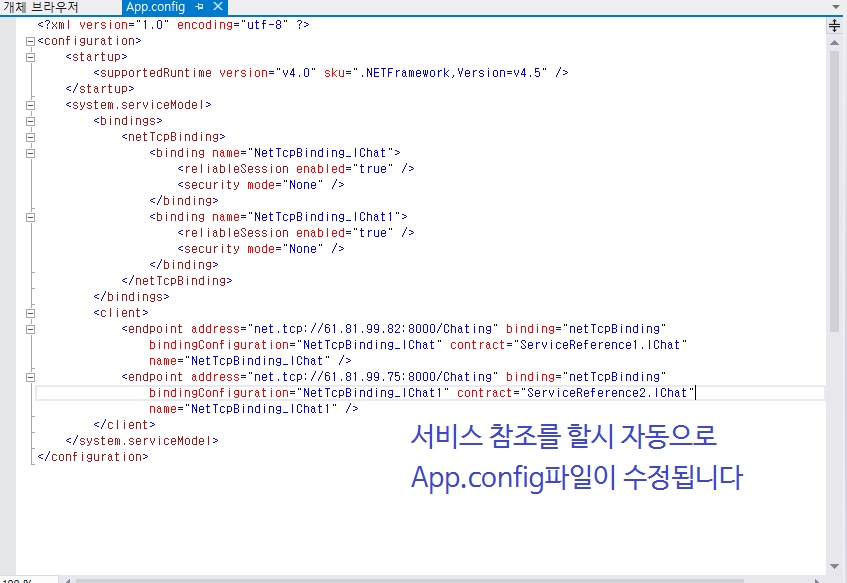
- 서비스 참조 : 프록시 객체 생성(**서비스 참조 추가시 서버가 열려 있어야 함)**

<http://192.168.0.9/GetService>

|  |  |
| --- | --- |
| C:\Users\Administrator\Desktop\zz\1번.png | **C:\Users\Administrator\Desktop\zz\3번.png** |

****

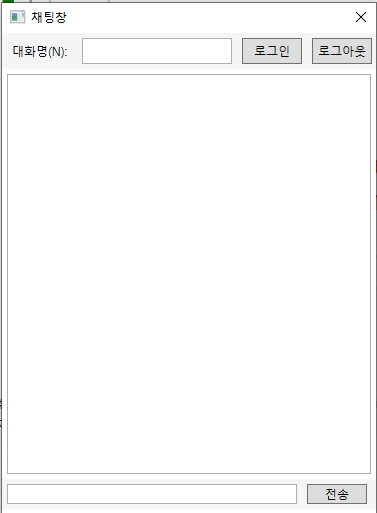
|  |
| --- |
| using System;  using System.Windows;  **using \_0427\_ChatClient.ServiceReference1;**  namespace \_0427\_ChatClient  {  /// <summary>  /// MainWindow.xaml에 대한 상호 작용 논리  /// </summary>  public partial class MainWindow : Window |

****

**App.config**

1. **Main화면 구성**

|  |
| --- |
| **메인 xaml 파일** |
| <Window x:Class="\_0427\_ChatClient.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:\_0427\_ChatClient"  mc:Ignorable="d"  Title="채팅창" Height="520" Width="390" ResizeMode="NoResize">  <StackPanel>  <!--첫번째 라인-->  <StackPanel Orientation="Horizontal" Background="WhiteSmoke">  <Label Margin="5">대화명(\_N):</Label>  <TextBox Name="nickname" Width="150" Margin="5" />  <Button Name="Join" Content="로그인" Margin="5" Width="60"/>  <Button Name="Leave" Content="로그아웃" Margin="5" Width="60"/>  </StackPanel>  <!--두번째 라인-->  <ListBox Name="chatlist" Height="400" Margin="5"/>  <!--세번째 라인-->  <StackPanel Orientation="Horizontal" Background="WhiteSmoke">  <TextBox Name="msgbox" Margin="5" Width="290"/>  <Button Name="send" Content="전송" Margin="5" Width="60"/>  </StackPanel>  </StackPanel>  </Window> |

****

1. **코드 구현**

|  |
| --- |
| **메인 cs파일** |
| using System;  using System.ServiceModel;  using System.Windows;  using \_0427\_ChatClient.ServiceReference1;  namespace \_0427\_ChatClient  {  /// <summary>  /// MainWindow.xaml에 대한 상호 작용 논리  /// </summary>  public partial class MainWindow : Window, IChatCallback  {  public string NickName  {  get { return nickname.Text; }  set { nickname.Text = value; }  }  public string Message  {  get { return msgbox.Text; }  set { msgbox.Text = value; }  }  private ChatClient chat = null;  public MainWindow()  {  InitializeComponent();  InstanceContext site = new InstanceContext(this);  chat = new ChatClient(site);  }  #region IChatCallback  public void Receive(string nickname, string message)  {  string msgtemp = string.Format("[{0}] {1}", nickname, message);  chatlist.Items.Add(msgtemp);  }  public void UserEnter(string nickname)  {  string msgtemp = string.Format("{0}님이 로그인하셨습니다.", nickname);  chatlist.Items.Add(msgtemp);  }  #endregion  public void UserLeave(string nickname)  {  string msgtemp = string.Format("{0}님이 로그아웃하셨습니다.", nickname);  chatlist.Items.Add(msgtemp);  }  #region 버튼 핸들러  private void Join\_Click(object sender, RoutedEventArgs e)  {  try  {  //서버 접속  if (chat.Join(NickName) == false)  UserEnter("로그인 실패");  }  catch (Exception ex)  {  UserEnter(string.Format("접속 오류 :{0}", ex.Message));  }  }  private void Leave\_Click(object sender, RoutedEventArgs e)  {  try  {  chat.Leave(NickName);  UserLeave(NickName);  }  catch (Exception ex)  {  UserEnter(string.Format("나가기 오류 :{0}", ex.Message));  }  }  private void send\_Click(object sender, RoutedEventArgs e)  {  try  {  chat.Say(NickName, Message);  msgbox.Clear();  }  catch (Exception)  {  MessageBox.Show("로그인을 먼저하세요");  }  }  }  #endregion  } |