**WCF - 채팅프로그램**

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

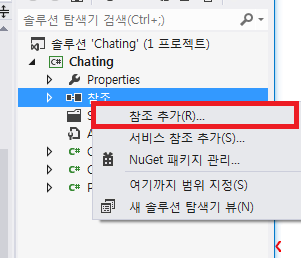
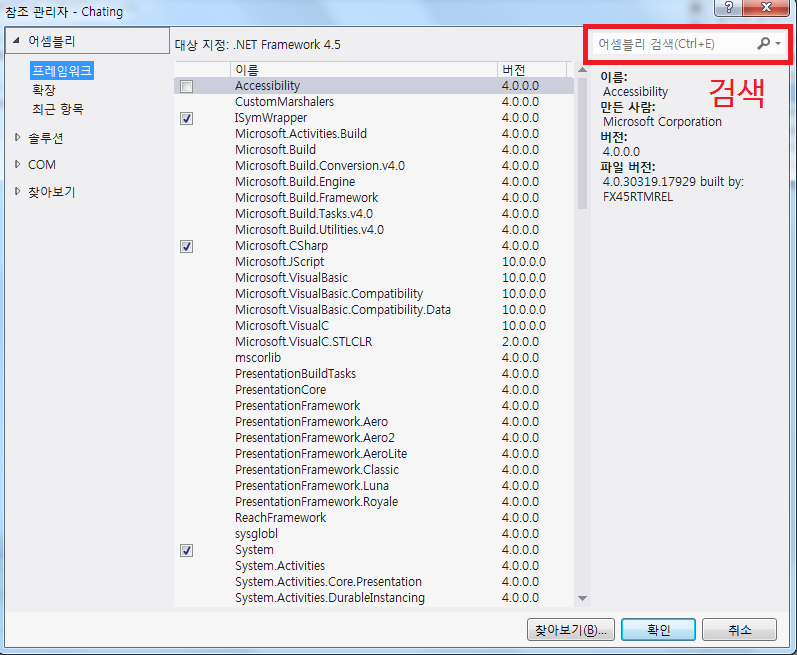
1. **프로젝트 생성 및 기본 어셈블리 추가**

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

프로젝트 생성 : 비주얼 스튜디오 – 콘솔 응용프로그램

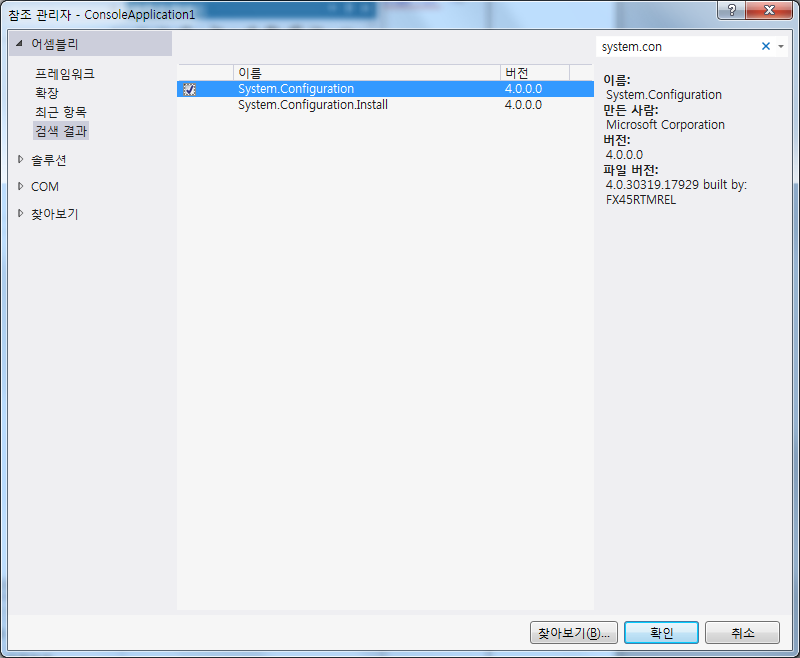
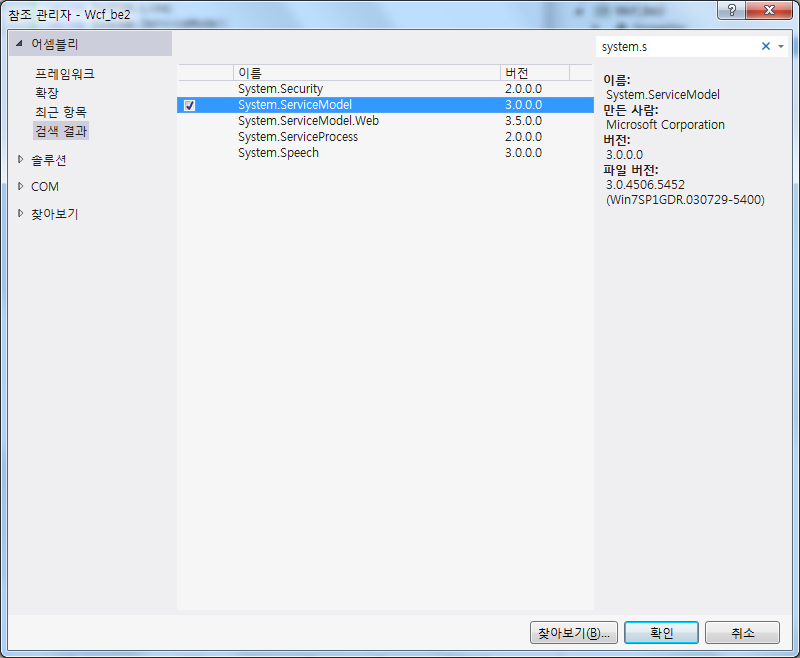
프로젝트 생성 후 참조추가 필수

**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**

1. **인터페이스 정의(1대 1통신과 동일)**

|  |
| --- |
| **이중계약을 위한 인터페이스 정의 : IChat, IChatCallback** |
| using System;  using System.ServiceModel;  namespace Chating  {  //서비스 계약 처리  [ServiceContract(SessionMode = SessionMode.Required, CallbackContract =typeof(IChatCallback))]  interface IChat  {  //첫번째 호출되어야 할 메서드(IsInitiating = true) : 세션이열림  //https://docs.microsoft.com/ko-kr/dotnet/api/system.servicemodel.operationcontractattribute.isinitiating?view=dotnet-plat-ext-5.0  [OperationContract(IsOneWay = false, IsInitiating = true, IsTerminating = false)]  bool Join(string idx);  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = false)]  void Say(string idx, string msg);  //마지막에 호출되어야 할 메서드(IsTerminating= true):세션이닫힘  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = true)]  void Leave(string idx);  }  public interface IChatCallback  {  [OperationContract(IsOneWay = true)]  void Receive(string idx, string message);  [OperationContract(IsOneWay = true)]  void UserEnter(string idx);  [OperationContract(IsOneWay = true)]  void UserLeave(string idx);  }  } |

1. **서비스 객체 정의**

|  |
| --- |
| **ChatService** |
| using System;  using System.Collections.Generic;  using System.ServiceModel;  using System.Text;  using System.Threading;  namespace \_1124WCFService  {  //델리게이트 정의  public delegate void Chat(string idx, string msg, string type);  class ChatService : IChat  {  //동기화 작업을 위해서 가상의 객체 생성[추가]  private static Object syncObj = new Object();  //채팅방에 있는 유저 이름 목록[추가]  private static List<string> Chatter = new List<string>();  //개인용 델리게이트  private Chat MyChat;  //1대 다 통신을 위한 클라이언트에 보낼 정보를 담고 있는 델리게이트[추가]  private static Chat List;  IChatCallback callback = null;  #region IChat 인터페이스  public bool Join(string idx)  {  MyChat = new Chat(UserHandler);  lock (syncObj) //동기화 된 상태에서 List에 추가  {  if (!Chatter.Contains(idx))  {  //2. 사용자에게 보내 줄 채널을 설정한다.  callback = OperationContext.Current.GetCallbackChannel<IChatCallback>();    //델리게이트에 추가  List += MyChat;  //현재 접속자 정보를 모두에게 전달  BroadcastMessage(idx, "", "UserEnter");  return true;  }  return false;  }  }  public void Leave(string idx)  {  List -= MyChat;  Chat d = null;  MyChat -= d;  BroadcastMessage(idx, "", "UserLeave");  }  public void Say(string idx, string msg)  {  BroadcastMessage(idx, msg, "Receive");  }  #endregion  private void BroadcastMessage(string idx, string msg, string msgType)  {  //1대 다 통신  //비동기로 UserHandler 함수 호출  if (List != null)  {  foreach (Chat handler in List.GetInvocationList())  {  handler.BeginInvoke(idx, msg, msgType, new AsyncCallback(EndAsync), null);  }  }  }  //클라이언트에 메시지 보내기  private void UserHandler(string idx, string msg, string msgType)  {  try  {  //클라이언트에게 보내기  switch (msgType)  {  //메시지 전송  case "Receive": callback.Receive(idx, msg); break;  //채팅 참여 알림  case "UserEnter": callback.UserEnter(idx); break;  //채팅 나감 알림  case "UserLeave": callback.UserLeave(idx); break;  }  }  catch//에러가 발생했을 경우  {  Console.WriteLine("{0} 에러", idx);  Leave(idx);  }  }  //비동기 결과 호출[기능은 없음]  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;  }  }  }  } |

1. **Config 파일 설정**

|  |
| --- |
| **APP.config 파일** |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <appSettings>  <add key="addr" value="net.tcp://localhost:8000/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="\_1124WCFService.ChatService" behaviorConfiguration="1111111" >  <host>  <baseAddresses>  <add baseAddress="http://localhost:9000/Chating" />  </baseAddresses>  </host>  <endpoint address="" binding="netTcpBinding" bindingConfiguration="DuplexBinding" contract="\_1124WCFService.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="1111111"> <!-- KEY -->  <serviceMetadata httpGetEnabled="true"/>  <serviceDebug includeExceptionDetailInFaults="true"/>  </behavior>  </serviceBehaviors>  </behaviors>  </system.serviceModel>  </configuration> |

1. **Hosting**

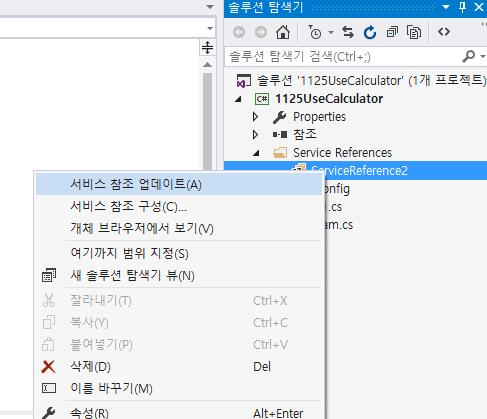
|  |
| --- |
| **메인 cs 파일** |
| using System;  using System.Collections.Generic;  using System.Configuration;  using System.Linq;  using System.ServiceModel;  using System.Text;  using System.Threading.Tasks;  namespace Chating  {  class Program  {  //Config파일을 이용하여 Hosting1코드 수정  static void Hosting3()  {  //System.Configuration 어셈블리 추가  Uri uri = new Uri(ConfigurationManager.AppSettings["addr"]);  //1. ServiceHost 객체(호스팅 객체)  //1) 서비스 객체  ServiceHost host = new ServiceHost(typeof(ChatService), uri);  host.Open();  Console.WriteLine("Press Any key to stop the service");  Console.WriteLine("address : " + uri.ToString());  Console.ReadKey(true);  //4. 호스트 종료  host.Abort();  host.Close();  }  static void Main(string[] args)  {  Hosting3();  }  }  } |

**\*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>**

* **기존 1대 1통신 클라이언트 프로그램을 서비스 참조 업데이트 하고 실행**
* **코드 수정 없음**

****