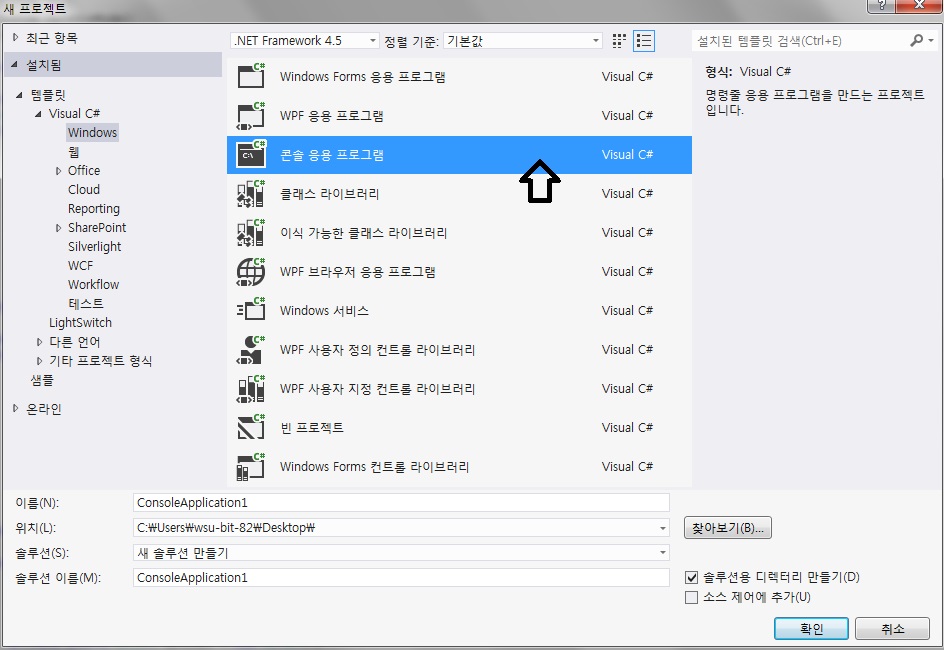
[ WCF TCP Binding 메신저 서비스 구현]

1. 새 프로젝트 생성  
   솔루션 : 콘솔 응용 프로그램  
   프로젝트명 : Chating  
   
2. 인터페이스 구현  
   파일명 : ChatService.cs (기본 Class)  
   네임 스페이스 추가 : using System.ServiceModel;

|  |
| --- |
| #region 1. 메세지 관련 Contract InterFace (클라이언트->서버)  [ServiceContract]  public interface IChat  {  [OperationContract(IsOneWay = false, IsInitiating = true, IsTerminating = false)]  string[] Join(string name);  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = false)]  void Say(string msg);  [OperationContract(IsOneWay = true, IsInitiating = false, IsTerminating = true)]  void Leave();  }  #endregion |

1. 필요한 변수 추가 및 구현 객체 구현

|  |
| --- |
| class ChatService : IChat  {  //델리게이트 선언  public delegate void Chat(string name, string msg, string type);  //동기화 작업을 위해서 가상의 객체 생성  private static Object syncObj = new Object();  //델리게이트 리스트 생성  private static Chat List;  //채팅방에 있는 유저 이름 목록  private static ArrayList Chatter = new ArrayList();  //키가됄 이름  private string name;  //개인 델리게이트  private Chat MyChat;  public string[] Join(string name)  {  MyChat = new Chat(UserHandler);  lock (syncObj)  {  if (!Chatter.Contains(name)) //이름이 기존 채터에 있는지 검색한다.  {  //이름과 이벤트를 추가한다.  this.name = name;  Chatter.Add(name);  //사용자에게 보내 줄 채널을 설정한다.  callback = OperationContext.Current.GetCallbackChannel<IChatCallback>();  //현재 접속자 정보를 모두에게 전달  BroadcastMessage(name, "", "UserEnter");  //델리게이터 추가  List += MyChat;  //사용자리스트를 보내준다.  string[] list = new string[Chatter.Count];  lock (syncObj)  {  Chatter.CopyTo(list);  }  return list;  }  else //이미 사용자가 사용하고 있는 이름일 경우  {  return null;  }  }  }  public void Say(string msg)  {  BroadcastMessage(this.name, msg, "Receive");  }  public void Leave()  {  if (this.name == null) return;  lock (syncObj)  {  Chatter.Remove(this.name);  }  List -= MyChat;  string msg = string.Format(this.name + "이가 나갔습니다");  BroadcastMessage(this.name, msg, "UserLeave");  }  } |

1. Callback 처리  
   4.1 Callback 인터페이스 등록  
   파일명 : ChatService.cs

|  |
| --- |
| #region 2. 클라이언트에 콜백할 CallBackContract (서버->클라이언트)  public interface IChatCallback  {  [OperationContract(IsOneWay = true)]  void Receive(string senderName, string message);  [OperationContract(IsOneWay = true)]  void UserEnter(string name);  [OperationContract(IsOneWay = true)]  void UserLeave(string name);  }  #endregion |

4.2 이중 계약에 두 개의 인터페이스 연결 : 기존 인터페이스의 callbackcontrack 속성을 추가  
파일명 : ChatService.cs

|  |
| --- |
| #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)]  string[] Join(string name); |

4.3 구현 클래스에 콜백 참조 추가  
클래스 : ChatService

|  |
| --- |
| private static ArrayList Chatter = new ArrayList();  //키가됄 이름  private string name;  //개인 델리게이트  private Chat MyChat;  IChatCallback callback = null; //🡸 추가 |

1. 메시지 전송을 위한 함수들 구현  
   클래스 : ChatService  
   5.1 함수 이름 : UserHandler  
    기능 : 사용자가 메시지를 전송해오는 방식을 설정하여 전송 타입에 맞게 데이터 전송

|  |
| --- |
| private void UserHandler(string name, string msg, string msgType)  {  try  {  //클라이언트에게 보내기  switch (msgType)  {  case "Receive":  callback.Receive(name, msg);  break;  case "UserEnter":  callback.UserEnter(name);  break;  case "UserLeave":  callback.UserLeave(name);  break;  }  }  catch//에러가 발생했을 경우  {  Leave();  }  } |

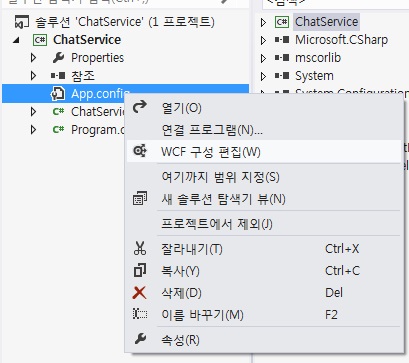
5.2 함수 이름 : BroadCastMessage  
 기능 : 접속한 모든 사용자들에게 메시지를 보내준다

|  |
| --- |
| private void BroadcastMessage(string name, string msg, string msgType)  {  if (List != null)  {  //현재 이벤트들을 전달한다.  foreach (Chat handler in List.GetInvocationList())  {  handler.BeginInvoke(name, msg, msgType, new AsyncCallback(EndAsync), null);  }  }  } |

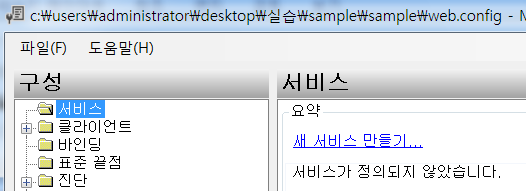
* 1. 함수 이름 : EndAsync  
     기능 : 콜백 함수로써 비동기식으로 메시지를 전송한 후 쓰레드 풀링에서 제외시킴

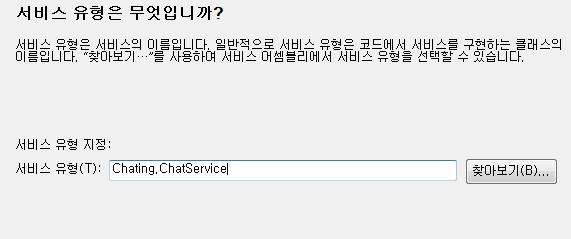
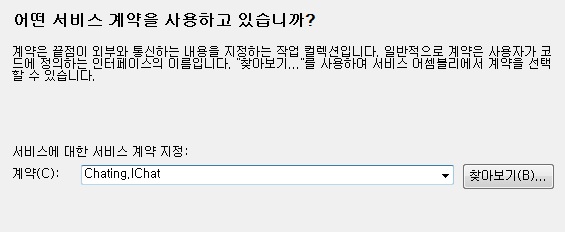
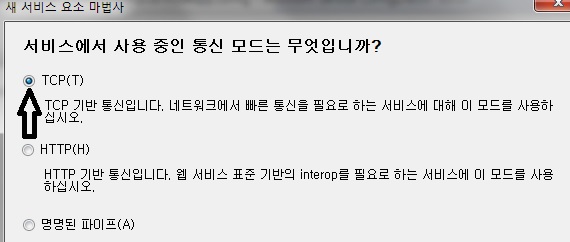
|  |
| --- |
| private void EndAsync(IAsyncResult ar)  {  Chat d = null;  try  {  System.Runtime.Remoting.Messaging.AsyncResult asres = (System.Runtime.Remoting.Messaging.AsyncResult)ar;  d = ((Chat)asres.AsyncDelegate);  d.EndInvoke(ar);  }  catch  {  List -= d;  }  } |

1. Binding 설정
   1. Binding 설정  
      구성편집 도구 이용



* 1. 서비스 >> 새 서비스 만들기



* 1. 서비스 유형 설정  
     유형 : Chating.ChatService  
     
  2. 서비스 계약  
     계약 : Chating.IChat  
     
  3. 통신 모드 선택  
     
  4. 주소 선택 : 미지정 후 완료!
  5. 생성된 config 파일

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <startup>  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />  </startup>  <system.serviceModel>  <services>  <service name="Chating.ChatService"  <endpoint address="" binding="netTcpBinding" bindingConfiguration=" "  contract="Chating.IChat" />  </service>  </services>  </system.serviceModel>  </configuration> |

* 1. 솔루션 탐색기에 App.config 파일에 AppSetting 세션 추가  
     \* console 메인에서 tcp주소값을 가져오기 위해서 추가

|  |
| --- |
| <?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://localhost:7878/Chating"/>**  **</appSettings>**  <system.serviceModel>  <services>  <service name="Chating.ChatService"  <endpoint address="" binding="netTcpBinding" bindingConfiguration=" "  contract="Chating.IChat" />  </service>  </services>  </system.serviceModel>  </configuration> |

6.9 서비스 참조로 가져갈수 있도록 baseAddress 추가

|  |
| --- |
| <?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://localhost:7878/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="Chating.ChatService" >  **<host>**  **<baseAddresses>**  **<add baseAddress="http://localhost:9000/GetService"/>**  **</baseAddresses>**  **</host>**  <endpoint address="" binding="netTcpBinding" bindingConfiguration=" "  contract="Chating.IChat" />  </service>  </services>  </system.serviceModel>  </configuration> |

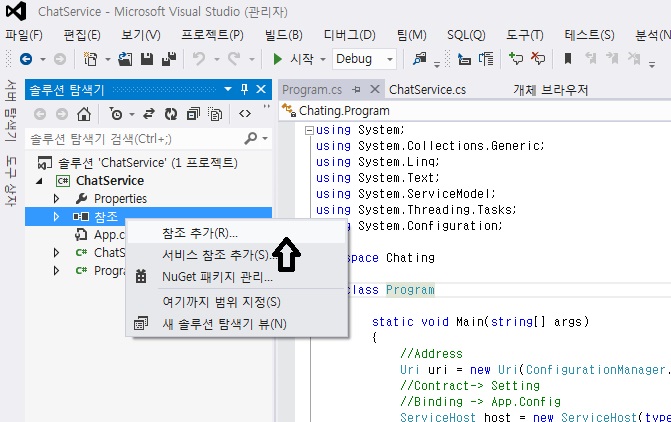
6.10 Binding 형태 설정 및 Binding 세부내용 설정

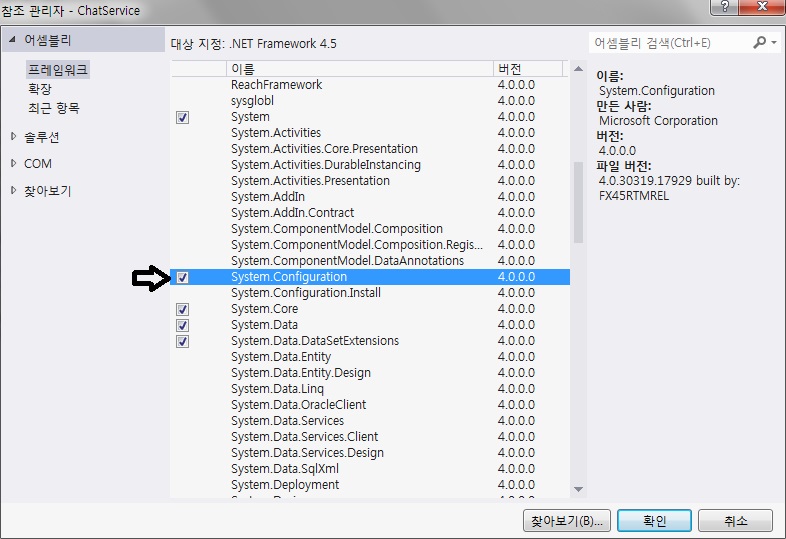
|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <startup>  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />  </startup>  <appSettings>  <add key="addr" value="net.tcp://localhost:7878/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="Chating.ChatService"  <host>  <baseAddresses>  <add baseAddress="http://localhost:9000/GetService"/>  </baseAddresses>  </host>  <endpoint address="" binding="netTcpBinding" bindingConfiguration=" **DuplexBinding** "  contract="Chating.IChat" />  </service>  </services>  **<bindings>**  **<netTcpBinding>**  **<binding name="DuplexBinding" maxConnections="100">**  **<reliableSession enabled="true"/>**  **<security mode="None"/>**  **</binding>**  **</netTcpBinding>**  **</bindings>**  </system.serviceModel>  </configuration> |

6.11 메타 데이터를 받을수 있도록 behaviors 추가

|  |
| --- |
| <?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://localhost:7878/Chating"/>  </appSettings>  <system.serviceModel>  <services>  <service name="Chating.ChatService" **behaviorConfiguration="FileServiceBehavior">**  <host>  <baseAddresses>  <add baseAddress="http://localhost:9000/GetService"/>  </baseAddresses>  </host>  <endpoint address="" binding="netTcpBinding" bindingConfiguration=" "  contract="Chating.IChat" />  </service>  <bindings>  <netTcpBinding>  <binding name="DuplexBinding" maxConnections="100">  <reliableSession enabled="true"/>  <security mode="None"/>  </binding>  </netTcpBinding>  </bindings>  **<behaviors>**  **<serviceBehaviors>**  **<behavior name="FileServiceBehavior">**  **<serviceMetadata httpGetEnabled="True"/>**  **<serviceDebug includeExceptionDetailInFaults="true"/>**  **</behavior>**  **</serviceBehaviors>**  **</behaviors>**  </services>  </system.serviceModel>  </configuration> |

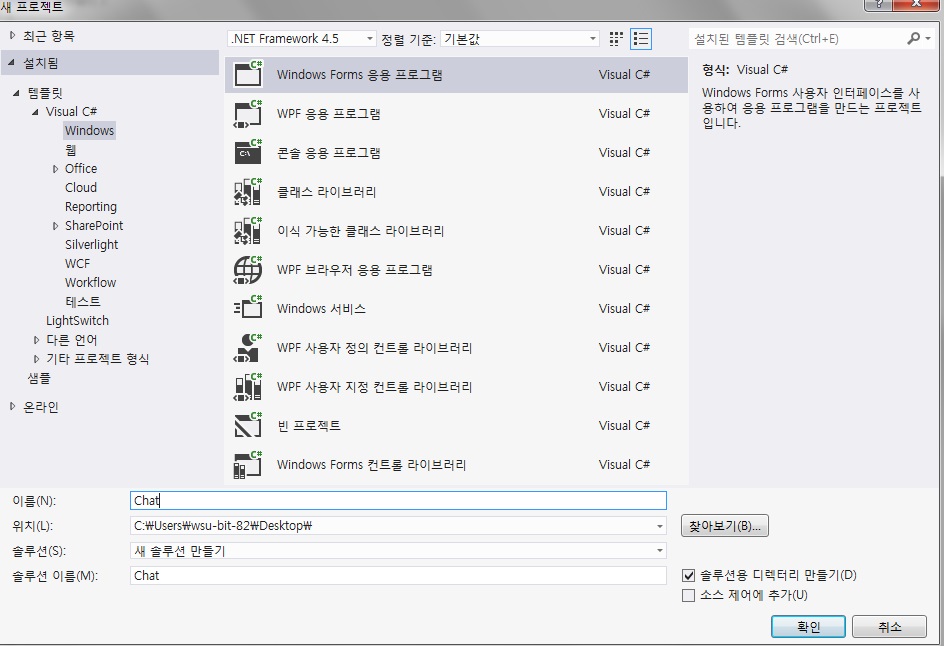
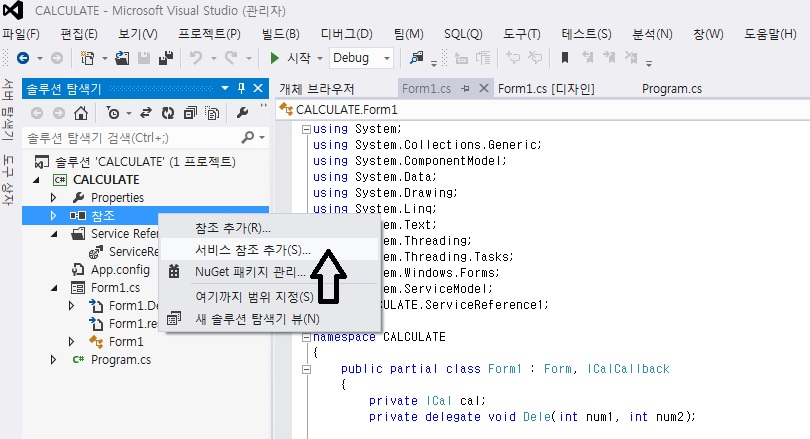
1. Main 함수 내부 구현  
   파일 : program.cs  
   네임스페이스 추가 : using System.ServiceModel;  
    using System.Configuration

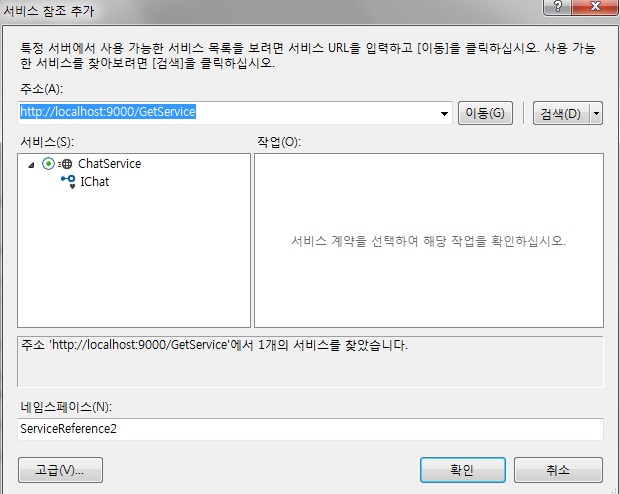
7.1 ConfigurationManager 사용을 위한 어셈블리 참조 추가  


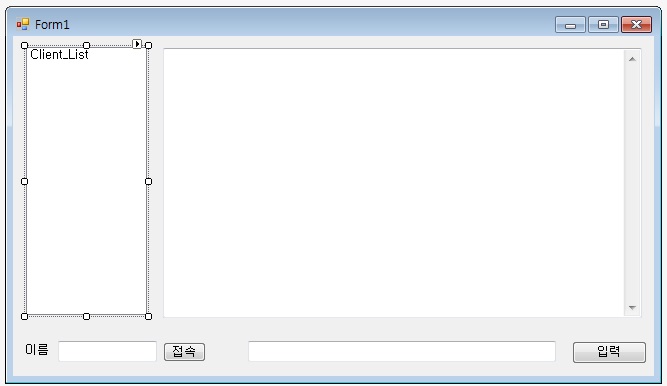


|  |
| --- |
| class Program  {  static void Main(string[] args)  {  //Address  Uri uri = new Uri(ConfigurationManager.AppSettings["addr"]);  //Contract-> Setting  //Binding -> App.Config  ServiceHost host = new ServiceHost(typeof(Chating.ChatService), uri);  //오픈  host.Open();  Console.WriteLine("채팅 서비스를 시작합니다. {0}", uri.ToString());  Console.WriteLine("멈추시려면 엔터를 눌러주세요..");  Console.ReadLine();  //서비스  host.Abort();  host.Close();  }  } |

[ WCF TCP Binding 메신저 클라이언트 구현]

1. 새 프로젝트 생성  
   솔루션 : 콘솔 응용 프로그램  
   프로젝트명 : Chat  
   
2. 서비스 참조 추가 (서비스 프로그램을 실행시킨뒤 참조 하도록한다)  
   주소 : http://localhost:9000/GetService  
   



1. UI 폼 구현  
   
2. 인터페이스 참조 ,Callback 인터페이스 상속   
   네임스페이스 추가 : using System.ServiceModel;  
    using Chat.ServiceReference1

|  |
| --- |
| namespace Chat  {  public partial class Form1 : Form, **IChatCallback**  {  private IChat Chat;  private string[] List;  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  InstanceContext site = new InstanceContext(this);  Chat = new ChatClient(site);  }  } |

1. IChatCallback 인터페이스 의 함수 구현

|  |
| --- |
| #region 서버에서 온 메세지 (서버->클라이언트)  public void Receive(string senderName, string message)  {  string msg = string.Format("{0} : {1}\r\n",senderName,message);  Text\_Chat.Text += msg;  }  public void UserEnter(string name)  {  string msg = string.Format(name + "님이 접속했습니다");  Text\_Chat.Text += msg;  Client\_List.Items.Add(name);  }  public void UserLeave(string name)  {  string msg = string.Format(name + "님이 퇴장했습니다");  Text\_Chat.Text += msg;  Client\_List.Items.Remove(name);  }  #endregion |

1. 접속 버튼 설정

|  |
| --- |
| private void Accept\_Click(object sender, EventArgs e)  {  List = Chat.Join(Text\_name.Text);  foreach (string member in List)  {  Client\_List.Items.Add(member);  }  Accept.Enabled = false;  } |

1. 입력 버튼 설정

|  |
| --- |
| private void Send\_Click(object sender, EventArgs e)  {  Chat.Say(Text\_msg.Text);  Text\_msg.Text = "";  } |

1. 폼이 종료되었을 때 종료 메시지를 위해 Closed 함수 구현

|  |
| --- |
| private void Form1\_FormClosed(object sender, FormClosedEventArgs e)  {  Chat.Leave();  } |