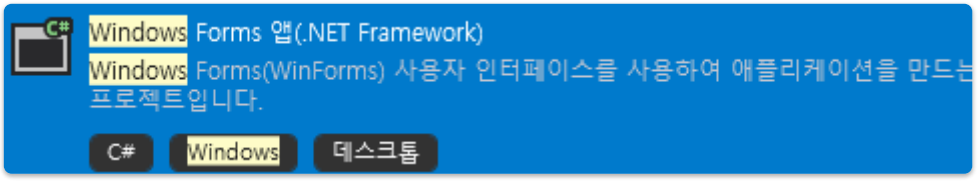
03. WCF 서비스 Client

#WCF #Client

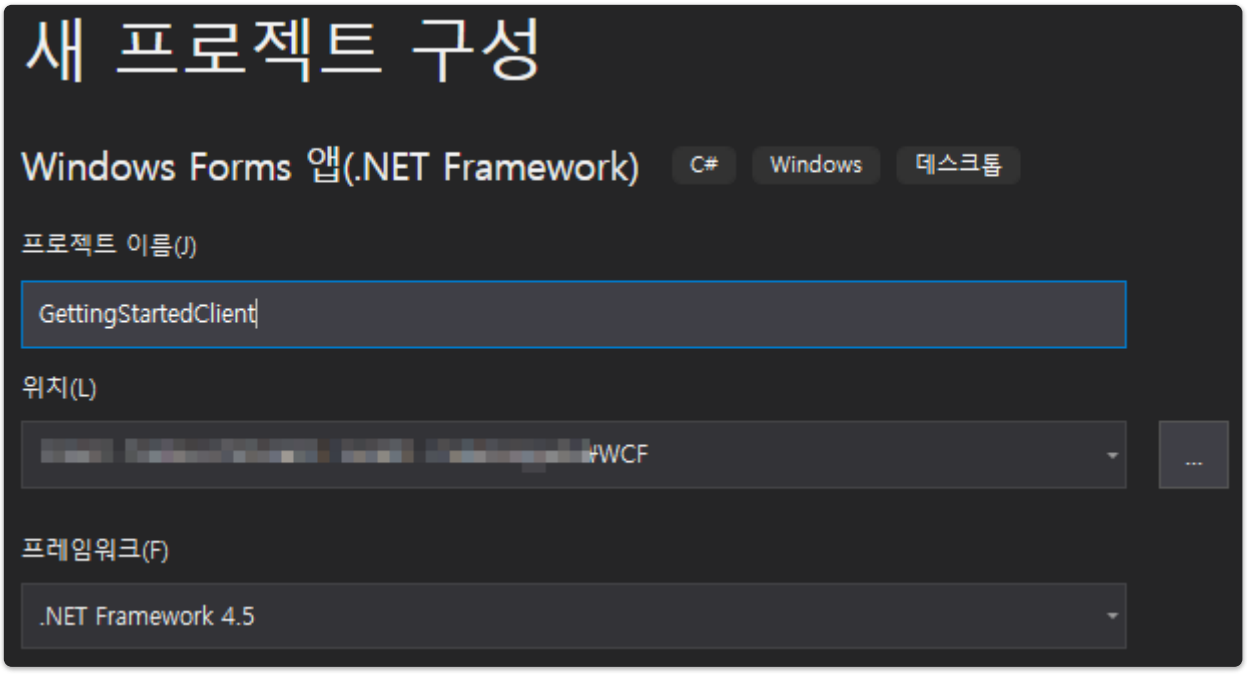
> Quote >
>
> https://learn.microsoft.com/ko-kr/dotnet/framework/wcf/how-to-host-and-run-a-basic-wcf-service

03.01. Project 생성

- .Net Framework → Windows Forms

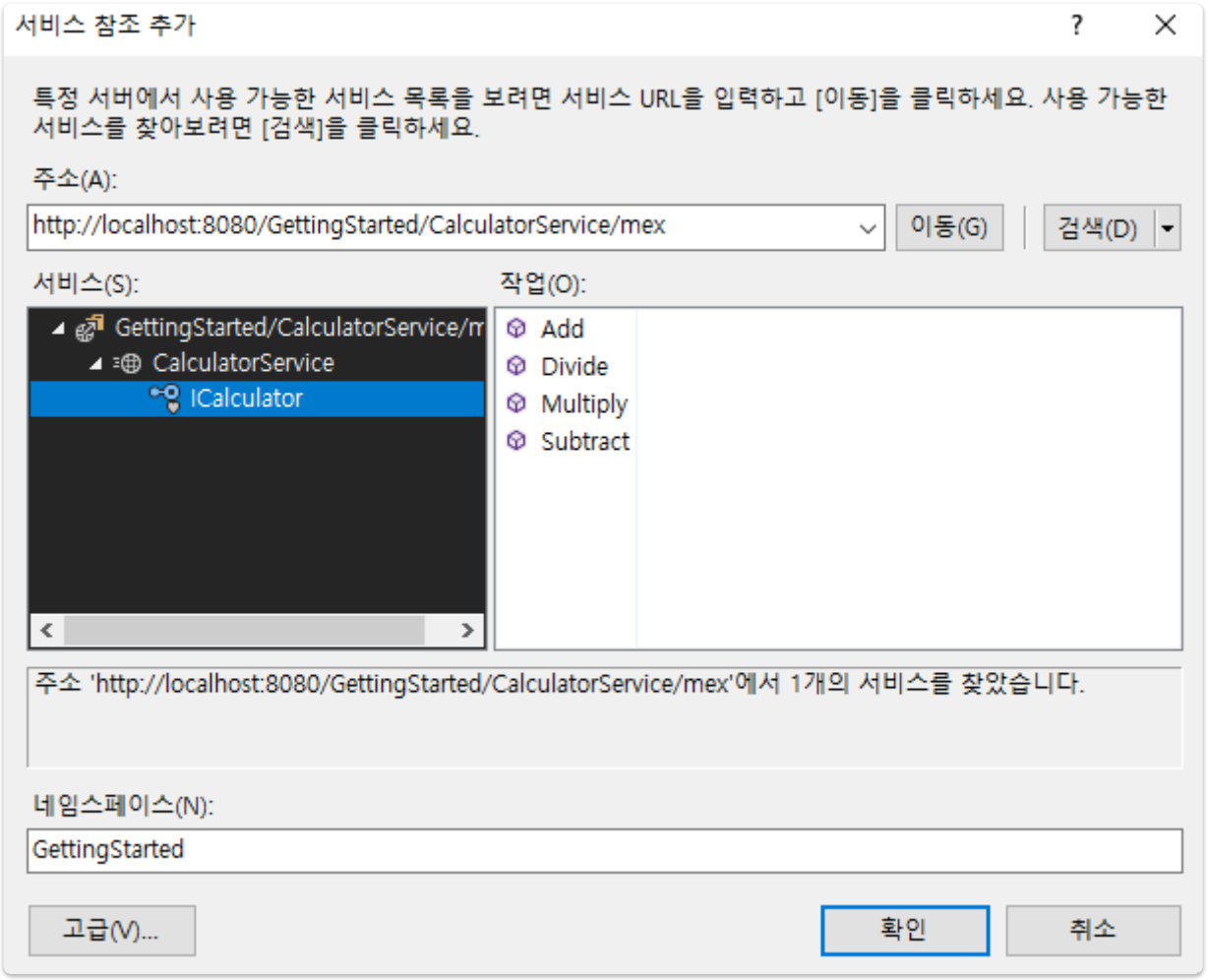


- 프로젝트 구성 정보 입력

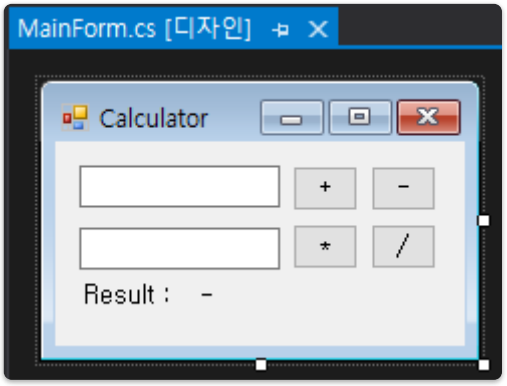


- 참조 추가
  - 어셈블리
    - 'ServiceModel'

- 서비스 참조 추가



03.02. 서비스 사용하는 코드 구현




```
public partial class MainForm : Form
{
        CalculatorClient client = new CalculatorClient();

        public MainForm()
        {
                InitializeComponent();
        }

        private void btn_add_Click(object sender, EventArgs e)
```

```
        {
                double result = client.Add(Double.Parse(edt_fir.Text),
Double.Parse(edt_sec.Text));
                lb_result.Text = result.ToString();
                Console.WriteLine($"Add({edt_fir.Text},{edt_sec.Text}) = {result}");
        }

        private void btn_sub_Click(object sender, EventArgs e)
        {
                double result = client.Subtract(Double.Parse(edt_fir.Text),
Double.Parse(edt_sec.Text));
                lb_result.Text = result.ToString();
                Console.WriteLine($"Add({edt_fir.Text},{edt_sec.Text}) = {result}");
        }

        private void btn_mul_Click(object sender, EventArgs e)
        {
                double result = client.Multiply(Double.Parse(edt_fir.Text),
Double.Parse(edt_sec.Text));
                lb_result.Text = result.ToString();
                Console.WriteLine($"Add({edt_fir.Text},{edt_sec.Text}) = {result}");
        }

        private void btn_div_Click(object sender, EventArgs e)
        {
                double result = client.Divide(Double.Parse(edt_fir.Text),
Double.Parse(edt_sec.Text));
                lb_result.Text = result.ToString();
                Console.WriteLine($"Add({edt_fir.Text},{edt_sec.Text}) = {result}");
        }

        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
                client.Close();
        }
}
```

03.03. ServiceModel Metadata 생성

> *Client에서 서비스 참조를 통해, 현재 호스팅 되고 있는 서비스에 대한 Metadata를 사용해도 되지만, 명시적으로 아래 유틸리티 도구를 통해 ServiceModel에 대한 Metadata를 생성하여 사용할 수도 있다.*

- https://learn.microsoft.com/ko-kr/dotnet/framework/wcf/servicemodel-metadata-utility-tool-svcutil-exe

```
svcutil.exe /language:cs /out:generatedProxy.cs /config:app.config
http://localhost:8080/Service/Calculator
```

- Command
  - svcutil.exe // C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools
  - /language:cs // or vb
  - /out:generatedProxy.cs // output file, 생성된 코드에 대한 파일 이름을 지정합니다.
  - /config:app.config // output file, 생성된 구성 파일의 파일 이름을 지정합니다.
  - http://localhost:8080/Service/Calculator // 서비스 URI
- app.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_ICalculator" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8080/Service/Calculator/CalculatorService"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_ICalculator"
                contract="ICalculator" name="BasicHttpBinding_ICalculator" />
        </client>
    </system.serviceModel>
</configuration>
```

- generatedProxy.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     이 코드는 도구를 사용하여 생성되었습니다.
//     런타임 버전:4.0.30319.42000
//
//     파일 내용을 변경하면 잘못된 동작이 발생할 수 있으며, 코드를 다시 생성하면
//     이러한 변경 내용이 손실됩니다.
// </auto-generated>
//------------------------------------------------------------------------------
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
[System.ServiceModel.ServiceContractAttribute(ConfigurationName="ICalculator")]
public interface ICalculator
{
```

```
[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Add",
ReplyAction="http://tempuri.org/ICalculator/AddResponse")]
    double Add(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Add",
ReplyAction="http://tempuri.org/ICalculator/AddResponse")]
    System.Threading.Tasks.Task<double> AddAsync(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Subtract
", ReplyAction="http://tempuri.org/ICalculator/SubtractResponse")]
    double Subtract(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Subtract
", ReplyAction="http://tempuri.org/ICalculator/SubtractResponse")]
    System.Threading.Tasks.Task<double> SubtractAsync(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Multiply
", ReplyAction="http://tempuri.org/ICalculator/MultiplyResponse")]
    double Multiply(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Multiply
", ReplyAction="http://tempuri.org/ICalculator/MultiplyResponse")]
    System.Threading.Tasks.Task<double> MultiplyAsync(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Divide",
ReplyAction="http://tempuri.org/ICalculator/DivideResponse")]
    double Divide(double n1, double n2);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/ICalculator/Divide",
ReplyAction="http://tempuri.org/ICalculator/DivideResponse")]
    System.Threading.Tasks.Task<double> DivideAsync(double n1, double n2);
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
public interface ICalculatorChannel : ICalculator, System.ServiceModel.IClientChannel
{
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
public partial class CalculatorClient : System.ServiceModel.ClientBase<ICalculator>,
ICalculator
{

    public CalculatorClient()
```

```
    {
    }

    public CalculatorClient(string endpointConfigurationName) :
            base(endpointConfigurationName)
    {
    }

    public CalculatorClient(string endpointConfigurationName, string remoteAddress) :
            base(endpointConfigurationName, remoteAddress)
    {
    }

    public CalculatorClient(string endpointConfigurationName,
System.ServiceModel.EndpointAddress remoteAddress) :
            base(endpointConfigurationName, remoteAddress)
    {
    }

    public CalculatorClient(System.ServiceModel.Channels.Binding binding,
System.ServiceModel.EndpointAddress remoteAddress) :
            base(binding, remoteAddress)
    {
    }

    public double Add(double n1, double n2)
    {
        return base.Channel.Add(n1, n2);
    }

    public System.Threading.Tasks.Task<double> AddAsync(double n1, double n2)
    {
        return base.Channel.AddAsync(n1, n2);
    }

    public double Subtract(double n1, double n2)
    {
        return base.Channel.Subtract(n1, n2);
    }

    public System.Threading.Tasks.Task<double> SubtractAsync(double n1, double n2)
    {
        return base.Channel.SubtractAsync(n1, n2);
    }

    public double Multiply(double n1, double n2)
    {
        return base.Channel.Multiply(n1, n2);
    }

    public System.Threading.Tasks.Task<double> MultiplyAsync(double n1, double n2)
    {
        return base.Channel.MultiplyAsync(n1, n2);
```

```
    }

    public double Divide(double n1, double n2)
    {
        return base.Channel.Divide(n1, n2);
    }

    public System.Threading.Tasks.Task<double> DivideAsync(double n1, double n2)
    {
        return base.Channel.DivideAsync(n1, n2);
    }
}
```