01. WCF 서비스 계약 인터페이스 정의 및 구현

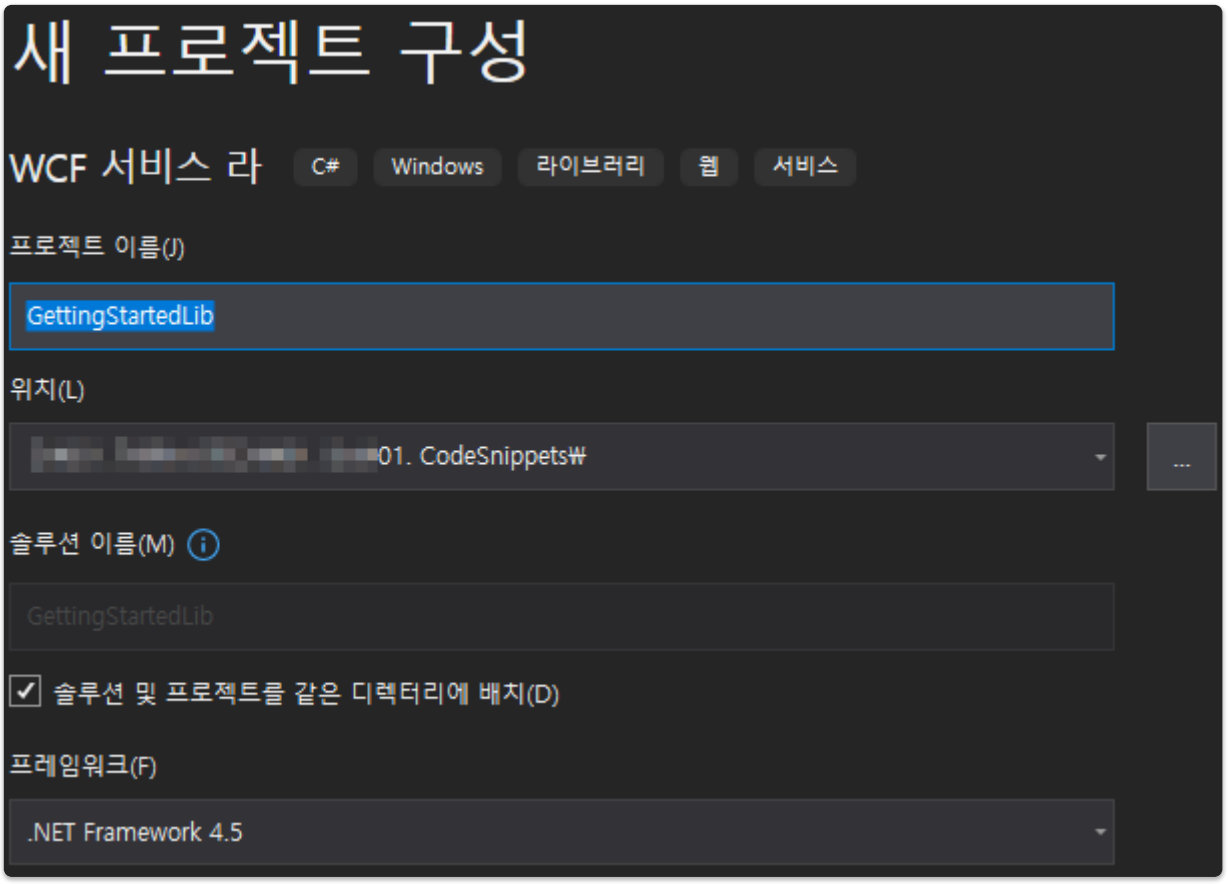
#WCF #Service

> Quote >
>
> https://learn.microsoft.com/ko-kr/dotnet/framework/wcf/how-to-define-a-wcf-service-contract
> https://learn.microsoft.com/ko-kr/dotnet/framework/wcf/how-to-implement-a-wcf-contract
> https://learn.microsoft.com/ko-kr/dotnet/framework/wcf/feature-details/configuring-system-provided-bindings

01.01. Project 생성

- WCF 서비스 라이브러리 선택

- 프로젝트 구성 정보 입력



- 구성 정보
  - IService.cs : 서비스 계약의 기본 정의
  - Service.cs : 서비스 계약의 기본 구현

- App.config : VS WCF 서비스 호스트 도구를 사용하여 기본 서비스를 로드하는 데 필요한 구성정보
  - IService1.cs → ICalculator.cs
  - Service1.cs → CalculatorService.cs
  - App.Config

01.02. 서비스 계약 인터페이스 정의

- IService.cs

```
namespace GettingStartedLib
{
    [ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples")]
    public interface ICalculator
    {
        [OperationContract]
        double Add(double n1, double n2);
        [OperationContract]
        double Subtract(double n1, double n2);
        [OperationContract]
        double Multiply(double n1, double n2);
        [OperationContract]
        double Divide(double n1, double n2);
    }
}
```

01.03 서비스 계약 구현

- Service.cs

```
namespace GettingStartedLib
{
    public class CalculatorService : ICalculator
    {
        public double Add(double n1, double n2)
        {
            double result = n1 + n2;
            Console.WriteLine("Received Add({0},{1})", n1, n2);
            // Code added to write output to the console window.
            Console.WriteLine("Return: {0}", result);
            return result;
        }

        public double Subtract(double n1, double n2)
        {
```

```
            double result = n1 - n2;
            Console.WriteLine("Received Subtract({0},{1})", n1, n2);
            Console.WriteLine("Return: {0}", result);
            return result;
        }

        public double Multiply(double n1, double n2)
        {
            double result = n1 * n2;
            Console.WriteLine("Received Multiply({0},{1})", n1, n2);
            Console.WriteLine("Return: {0}", result);
            return result;
        }

        public double Divide(double n1, double n2)
        {
            double result = n1 / n2;
            Console.WriteLine("Received Divide({0},{1})", n1, n2);
            Console.WriteLine("Return: {0}", result);
            return result;
        }
    }
}
```

- App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <!-- 서비스 라이브러리 프로젝트를 배포할 때 호스트의 app.config 파일에 구성 파일의 내용을 추가해야 합니다.
  System.Configuration이 라이브러리에 대한 구성 파일을 지원하지 않습니다. -->
  <system.serviceModel>
    <services>
```

```
  <service name="GettingStartedLib.CalculatorService">
   → 서비스 이름
    <endpoint address="" binding="wsHttpBinding" contract="GettingStartedLib.ICalculator">
    → 서비스 계약 인터페이스 정의 바인딩
```

```
      (wsHttpBinding 모드 사용시 보안 설정 필요!)
    -> 보안 설정 없이 원격 PC에서 사용할 수 있으려면 binding="basicHttpBinding" 설정
       -> 기본적으로 [BasicHttpBinding](https://learn.microsoft.com/ko-
kr/dotnet/api/system.servicemodel.basichttpbinding) 바인딩을 제외한 모든 바인딩에는 보안이 설정
되어 있습니다
```

```
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
        <host>
          <baseAddresses>
```

```
        <add baseAddress="http://localhost:8080/GettingStarted/CalculatorService" />
        → 서비스 기본 주소
```

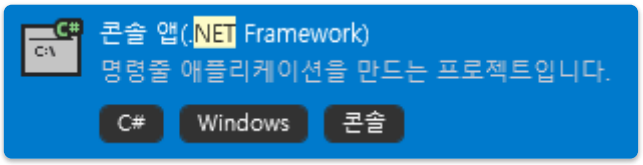
```
          </baseAddresses>
        </host>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!— 메타데이터 정보를 공개하지 않으려면
          배포하기 전에 아래 값을 false로 설정하십시오. -->
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>
          <!— 디버깅 목적으로 오류에서 예외 정보를 받으려면
          아래의 값을 true로 설정하십시오. 예외 정보를 공개하지 않으려면
         배포하기 전에 false로 설정하십시오. -->
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>

</configuration>
```

02. WCF 서비스 호스트

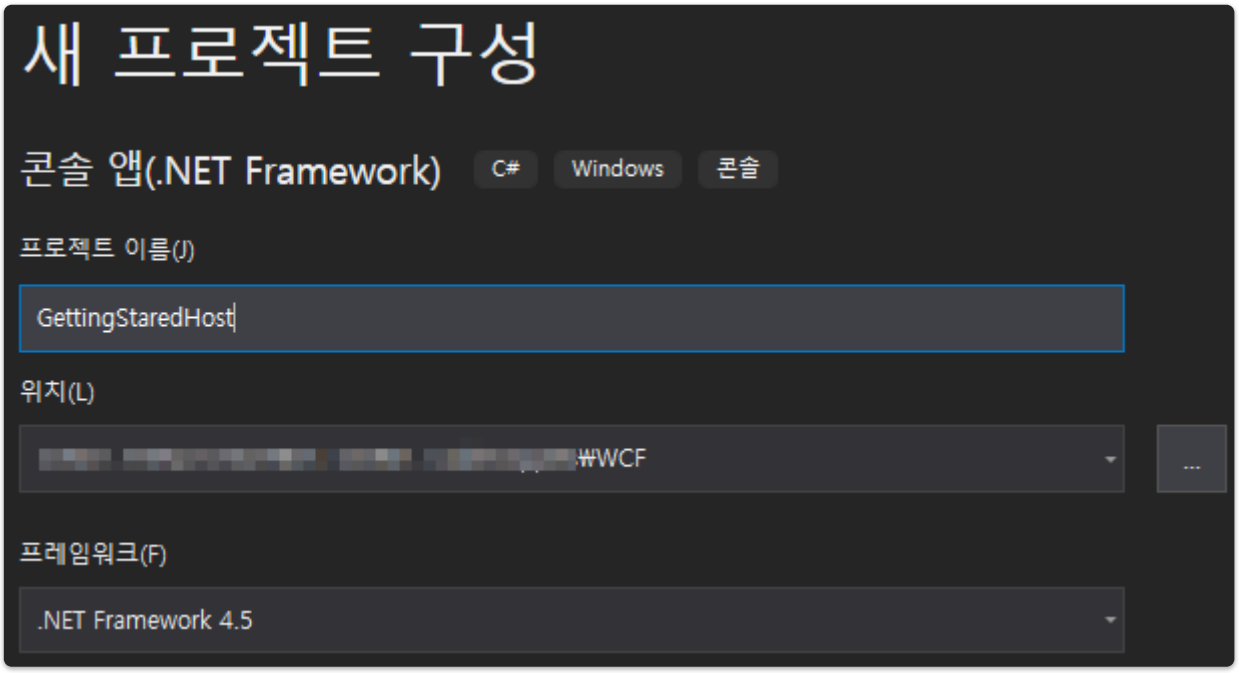
#host

02.01. Project 생성

- .Net Framework → Console App

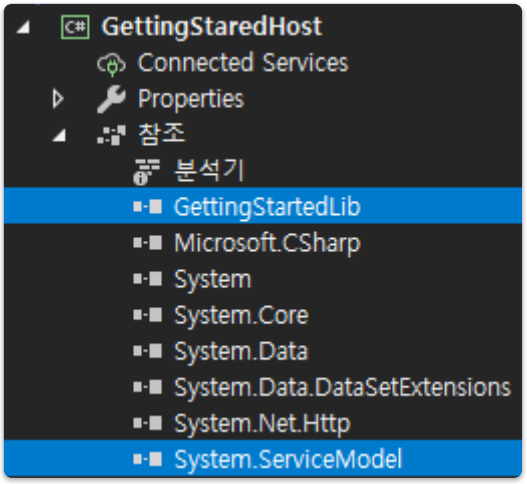


- 프로젝트 구성 정보 입력

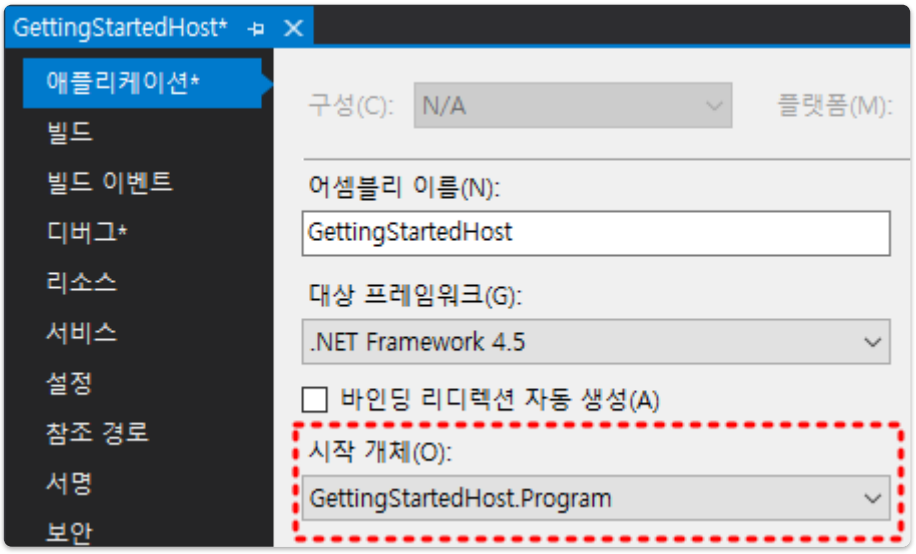


- 참조 추가
  - 솔루션
    - 'GettingStaredLib'
  - 어셈블리

    - 'ServiceModel'



- 시작 개체 설정



02.02. 서비스를 호스팅 하는 코드 구현

- Service Library - App.Config 에 baseAddresses에 바인딩 된 주소 사용

```
class Program
{
        static void Main(string[] args)
        {
                // Step 1: Create a URI to serve as the base address.
                Uri baseAddress = new
Uri("http://localhost:8080/GettingStarted/CalculatorService");

                ServiceHost selfHost = new ServiceHost(typeof(CalculatorService), baseAddress);

                try
                {
                        // Step 3: Add a service endpoint.
                        selfHost.AddServiceEndpoint(typeof(ICalculator), new WSHttpBinding(),
"CalculatorService");

                        // Step 4: Enable metadata exchange.
                        ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
                        smb.HttpGetEnabled = true;
                        selfHost.Description.Behaviors.Add(smb);

                        // Step 5: Start the service.
                        selfHost.Open();
                        Console.WriteLine("The service is ready.");

                        // Close the ServiceHost to stop the service.
                        Console.WriteLine("Press <Enter> to terminate the service.");
                        Console.WriteLine();
```

```
                        Console.ReadLine();
                        selfHost.Close();
                }
                catch (CommunicationException ce)
                {
                        Console.WriteLine("An exception occurred: {0}", ce.Message);
                        selfHost.Abort();
                }
        }
}
```