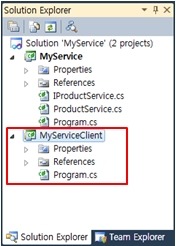
**WCF의 기본 개념 및 첫번째 실습2**

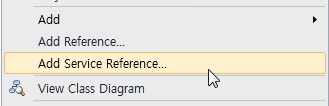
서비스를 구현하였으니, 이제 이 서비스를 이용할 수 있는 클라이언트를 구현해야 할 때입니다.

클라이언트도 Console 어플리케이션을 이용해보도록 하겠습니다.

1. 지난번 서비스를 만들었던 같은 솔루션내에 새로운 프로젝트를 추가합니다. (새로운 솔루션을 만들어도 상관은 없습니다 ^^) 아래 그림과 같이 저는 **"MyServiceClient"**라는 이름의 콘솔 어플리케이션 프로젝트를 추가 하였습니다.

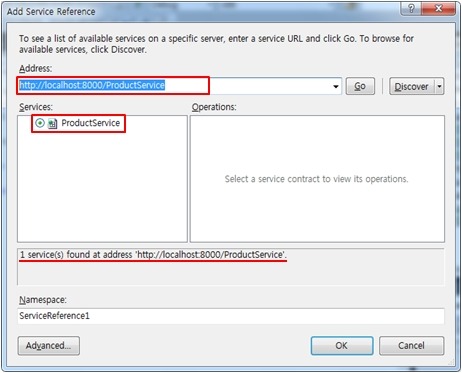
[](http://cfile9.uf.tistory.com/original/201188224B4B3B1B14D6F0)

2. 클라이언트 프로젝트에 지난번 만들었던 서비스를 참조 추가 해야합니다.

[](http://cfile25.uf.tistory.com/original/151F7E1D4B4B3EB28BB518)

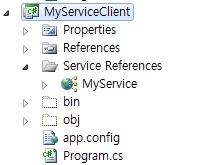
그러면, 서비스를 참조 추가할 수 있는 창이 뜹니다. 이 창의 Address 텍스트 박스에**"http://localhost:8000/ProductService"** 를 입력하고, "Go" 버튼을 클릭합니다. (방금 입력한 주소는 서비스 엔드포인트 주소입니다. 기억나시죠??) 이렇게 하면~ 서비스를 찾는 메시지가 뜨고 얼마 뒤 다음 그림과 같이 서비스를 찾았다는 메시지가 뜨는 것을 확인할 수 있습니다.

참~!! 이때, 아주 당연한 얘기지만, 서비스가 실행 중이어야 합니다. ctrl + F5 를 눌러 서비스를 실행하는 콘솔 창을 띄워놓은 상태에서 서비스를 찾아야 합니다.

[](http://cfile1.uf.tistory.com/original/15404A134B4B42E72A07B7)

Namespace 텍스트 박스엔 "ServiceReference1"이라고 적혀있지만, 이는 개발자 마음대로 변경 가능합니다. 그래서, 저는 "**MyService**"라고 수정 후에 "OK" 버튼을 클릭하였습니다.

이 작업을 모두 완료하면 Solution Explorer의 모습이 다음 그림과 같이 조금 바뀌게 됩니다.

[](http://cfile21.uf.tistory.com/original/142243164B4B44E7078B3C)

**"Service Reference"**라는 폴더가 새로 생겼고, 그 밑에 방금 등록했던 **"MyService"**가 추가 되어있는 것을 확인할 수 있습니다.

이로써, 클라이언트에서 서비스를 사용할 준비는 거의 끝났습니다.

|  |
| --- |
| [비주얼스튜디오가 자동으로 해주는 일]  서비스와 통신할 수 있는 proxy 클래스를 생성 해 주는 것입니다. 이는 예전 닷넷 웹서비스를 참조 했을 때, proxy 클래스를 생성해 주던 것과 아주 비슷합니다. |

|  |
| --- |
| [Metadata Exchange Endpoint]  비주얼스튜디오에서 쉽게 서비스를 참조 추가할 수 있었던 이유, 바로 **Metadata Exchange Endpoint** 입니다.  **WCF의 Metadata**라 함은, **WCF 서비스와 통신하는 방식을 설명한 것**이라고 간단하게 생각하시면 됩니다. 그러면 MEX(Metadata Exchange Endpoint) 가 무엇인지 대충 감이 오시나요?  네, 추측하신대로 이러한 메타데이터를 교환하기 위한 엔드포인트를 말합니다. 클라이언트에서 이 MEX로 요청 메시지(request message)를 보내면, **MEX 에서는 서비스가 가지고 있는 엔드포인트에 대한 내용과 메시지의 포맷에 대한 내용을 WSDL 형태로 클라이언트에 전송**해 주는 것입니다.  비주얼 스튜디오에서 "서비스 참조 추가"를 실행하였을 때, 프록시 클래스와 config 파일을 생성해 주는 것은 모두 MEX에서 보내준 WSDL을 이용한 것이죠~  그럼, MEX를 정의하지 않으면 WCF 서비스와 통신하지 못할까요?  꼭 그렇지만은 않습니다. 물론, 비주얼 스튜디오에서 지원해주는 "서비스 참조 추가"를 이용하진 못합니다.  하지만,**svcutil.exe** 라는 명령어를 이용하여 프록시 클래스를 생성해 줄 수 있고, 이를 통해 서비스와 통신할 수 있습니다. |

3. 코드 입력

using MyServiceClient.MyService;

namespace MyServiceClient  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            ProductServiceClient proxy = new ProductServiceClient();  
            string strResult = proxy.GetFirstName("WCF");  
            Console.WriteLine(strResult);  
        }  
    }  
}

위의 코드는 클라이언트 Console 어플리케이션의 Program.cs 파일 내용입니다.

using 구문을 이용하여 포함시킨 **MyServiceClient.MyService 네임스페이스**는 비주얼 스튜디오에 의해 생성된 프록시 클래스의 네임스페이스입니다.

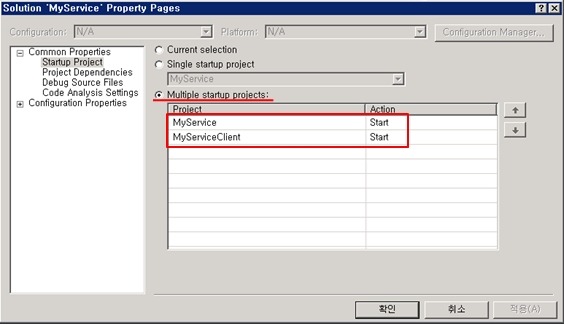
이 네임스페이스에서 상위 네임스페이스는 서비스에 정의 된 네임스페이스에 "Client"가 붙은 형태로 생성되며, 하위 네임스페이스는 서비스 참조 추가를 할 때 입력하여 준 것과 동일한 형태로 생성되어 집니다.

그리고, **ProductServiceClient 클래스**는 비주얼 스튜디오에 의해 생성된 프록시 클래스 입니다. 이 클래스를 이용하면 서비스에 정의되어 있는 메소드들을 클라이언트에서 맘껏 사용할 수 있게 되는 것입니다.

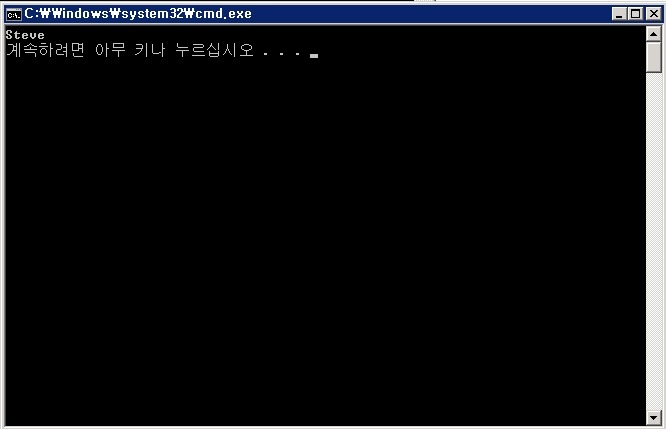
그럼, 실행을 한번 시켜 보도록 하죠~

현재, 솔루션에서 서비스 프로젝트와 클라이언트 프로젝트가 모두 동작해야 되기 때문에 솔루션의 속성을 조금 변경할 필요가 있습니다.

아래 그림에서 볼 수 있듯이, 솔루션 속성 창에서 "Multiple startup projects"를 선택하고 두 개의 프로젝트 모두 "Start" 하도록 값을 변경하여 줍니다.

[](http://cfile6.uf.tistory.com/original/1224C9034B4DB2CB034449)

이제 실행만 하면 됩니다. 과감하게~ F5 키를 눌러 실행을 시켜 보죠~

[](http://cfile4.uf.tistory.com/original/1731D1014B4DB34F817664)