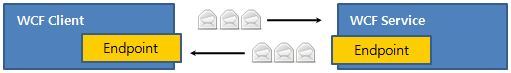
**WCF의 기본 개념 및 첫번째 실습1**

WCF가 웹 서비스와 비슷하게, **분산 어플리케이션**을 쉽게 개발할 수 있는 .NET 기술이라고 설명했었습니다. 분산 어플리케이션은 특정 기능을 수행하는 서버 프로그램과 그리고 이 서버 프로그램을 호출하는 클라이언트로 이루어져 있는 이러한 구조를 가집니다. (쉽게 웹 서비스를 사용하는 어플리케이션 구조를 생각하면 될 것 같습니다.)

물론, WCF도 예외는 아닙니다. WCF는 크게 **클라이언트**와 서버 프로그램, 그러니깐 **WCF 서비스**로 나눌 수 있습니다. (이제부터는 서버 쪽 WCF 프로그램을 WCF 서비스(또는 서비스)라고 지칭하겠습니다.)   
  
이러한 구조에서 클라이언트와 서비스는 메세지를 주고 받기 위해 아래 그림과 같이 **Endpoint** 를 제공합니다.

[](http://cfile4.uf.tistory.com/original/142D49254B43E5FF4D3582)

**구현 순서**

1. 서비스 구현
2. 서비스 호스팅
3. 클라이언트 구현
4. **WCF 서비스 구현**

**EndPoint**

endpoint는 WCF 에서 새롭게 정의되어진 단어로서 서비스에 접근할 수 있는 **interface**라 생각하면 될 것 같습니다.   
  
엔드포인트는 **Address, Binding, Contract** 세가지 요소로 이루어집니다. (이를 **"WCF 서비스의 ABC"** 라고 기억하시면 됩니다.)

그럼, 각 요소들에 대해 좀 더 자세한 설명을 해보겠습니다.

* **Address (Where)**

Address는 말 그대로 주소입니다. 서비스가 어디에 있는지에 대한 정보를 나타내는 것으로 메세지가 보내어져야 하는 곳을 말합니다. 메세지 전송을 위해 HTTP 프로토콜을 사용한다면 address는 "<http://myserver/myservice/>" 와 같은 모습을 할 것이며, 만약, TCP 프로토콜을 사용한다면 "net.tcp://myserver:8080/myservice" 와 같이 표현될 수 있습니다.

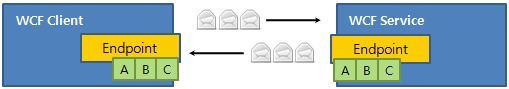
* **Binding (How)**

 Binding은 Channel을 정의합니다. Channel이란 메시지를 주고 받기 위한 여러 프로토콜을 명세하는 것을 말하는데, 여기에는 메시지를 전송하기 위한 전송 프로토콜, 보안의 적용 여부 그리고, 트랜잭션의 지원 여부 등이 포함됩니다. 전송 프로토콜의경우 HTTP, TCP, Named Pipes, MSMQ 등을 지원하여 서비스의 목적이나 환경에 따라서 적절하게 사용할 수 있다는 장점이 있습니다.

* **Contract (What)**

 Contract는 서비스의 operation에 필요한 메시지의 포맷을 정의합니다. 좀 더 쉽게 설명하자면, 메시지의 포맷이란 결국 서비스에서 제공하는 메소드와 이 메소드의 매개변수를 정의한 것입니다.(서비스가 제공하는 기능들을 정의한 것이라고 볼 수도 있겠네요) 실제로, 이후에 WCF 서비스를 만들어보면 알겠지만 이 Contract에는 서비스가 제공하는 메소드를 정의한 인터페이스가 매핑되는 것을 볼 수 있을 것입니다.

위의 그림을 조금 발전시킨 그림입니다. WCF 서비스와 클라이언트는 특수한 상황이 아닌 경우 양방향 통신을 하기 때문에 클라이언트 역시 엔드포인트를 가지고 있어야 합니다. 물론, 서비스와 클라이언트의 엔드포인트는 각각 A,B,C를 정의해야 겠죠~!!

[](http://cfile27.uf.tistory.com/original/170CC6264B43E6146CEFA0)

1. **서비스 호스팅**

서비스를 개발하고, 엔드포인트까지 정의가 끝이 나고 나면, 이제 이 서비스를 **호스팅(Hosting)**해야 합니다. 마지막으로 호스팅 작업을 해줘야 서비스는 외부에서 접근할 수 있는 상태가 됩니다.

WCF는 서비스를 호스팅 하기 위해 여러 가지의 방법을 제공합니다.

* 가장 보편적인 방법 : IIS Hosting
* Windows Service를 이용한 Hosting
* 윈폼 또는 콘솔 어플리케이션을 이용해서 호스팅

|  |
| --- |
| [기존 웹 서비스와의 차이점]  기존 웹 서비스의 경우에는 항상 IIS를 사용해야 했었고, 그 덕분에 http 프로토콜 만을 이용할 수 밖에 없었죠. 하지만, WCF 서비스는 여러 방식을 통해 호스팅 할 수 있는 장점을 가짐으로써, 적시적소에 서비스를 적용할 수 있는 큰 장점을 제공해 주는 것입니다. 더군다나, 이러한 것들을 개발자가 쉽게 개발할 수 있도록 해주었다는 것 역시 WCF의 큰 장점입니다. |

일단, 기본적인 사항은 여기까지 설명을 하구요, 나머지는 실제로 간단한 WCF 서비스를 만들어보면서 추가적으로 설명을 하겠습니다.  
다음으로 넘어가기 전에, 아래와 같은 내용을 다시 한번 되새김질(?) 하고 넘어가도록 하죠~

﻿서비스를 개발하기 위해선,,

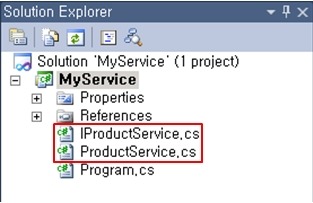
* 서비스가 제공하는 기능들을 정의 한 Contract(계약) 가 존재해야 한다.
* 서비스에 접근하기 위한 엔드포인트가 필요하며, 엔드포인트는 Address, Binding, Contract 로 구성되어 있다.
* 서비스를 노출하기 위해 서비스를 호스트 해야 하며, WCF는 호스팅을 수행하기 위한 여러 가지 방식을 제공해 준다.

**간단한 WCF 서비스 개발하기(실습)**

앞에서도 설명을 했었지만, WCF 서비스는 호스팅을 위한 여러 방법이 존재합니다. 처음으로 만드는 이 WCF 서비스는 여러 방법 중 **Console 어플리케이션을 이용한 방법**으로 간단하게 만들어볼까 합니다.

1. 우선, VS2010에서 Console Application 프로젝트를 생성합니다.

2. 그리고, 프로젝트에 아래 그림과 같이 각각 **IProductService.cs**, **ProductService.cs** 라는 이름을 가진  두 개의 파일을 추가 하였습니다.

[](http://cfile24.uf.tistory.com/original/1525EB264B45B8C420E7DB)

|  |  |
| --- | --- |
| IProductService | 인터페이스며, **서비스의 계약**을 정의합니다.  서비스의 계약은 한마디로 서비스가 제공하는 여러 메소드를 정의하는 인터페이스라고 생각하시면 됩니다. |
| ProductService | IProductService를 상속받아 실제 메소드를 구현하는 클래스입니다. |

3. 파일을 추가한 후에, 프로젝트에 **System.ServiceModel.dll** 어셈블리를 참조 추가합니다.  
(System.ServiceModel.dll 에 WCF 서비스를 위한 대부분의 클래스들이 포함되어 있습니다.)

4. 구현

먼저, IProductService 인터페이스 부터 구현해보겠습니다.(서비스 계약)

[ServiceContract]  
interface IProductService{  
 [OperationContract]  
 string GetFirstName(string empID);  
}

서비스 계약을 정의하는 방법

* 클래스에 **"ServiceContract"** 특성을 지정
* 메소드에는 **"OperationContract"** 특성을 지정

다음으로 ProductService 클래스를 구현해보겠습니다.

class ProductService: IProductService{  
    public string GetFirstName(string empID)    {  
        return"Steve";  
    }  
}

5. 호스팅하기(Console 어플리케이션을 이용한 호스팅)

Console 어플리케이션을 이용하여 호스팅을 하기 위해서 Program.cs 에 관련 코드를 다음과 같이 추가 하였습니다.

using System.ServiceModel;  
using System.ServiceModel.Description;  
namespace MyService

{  
 class Program

{  
 static void Main(string[]args)

{  
 Uri baseUri = new Uri("http://localhost:8000/ProductService");  
 using(ServiceHost serviceHost = new ServiceHost(typeof(ProductService), baseUri))

{  
 //Add Service Endpoint  
 serviceHost.AddServiceEndpoint(typeof(IProductService), new BasicHttpBinding(), "");  
   
               //Add Metadata Behavior  
 ServiceMetadataBehavior mexBehavior = new ServiceMetadataBehavior();  
 mexBehavior.HttpGetEnabled = true;  
 serviceHost.Description.Behaviors.Add(mexBehavior);

//Add Metadata Exchange Endpoint  
 serviceHost.AddServiceEndpoint(typeof(IMetadataExchange),   
                                              MetadataExchangeBindings.CreateMexHttpBinding(),"mex");  
 serviceHost.Open();

Console.WriteLine("Press <ENTER> to terminate WCFService.\n");  
 Console.ReadLine();  
 }  
        }  
    }  
}

이 코드에서의 핵심은 **ServiceHost** 클래스입니다. ServiceHost 객체를 이용하면, WCF 서비스를 호스팅 할 수 있다는 사실을 마음 속 아니, 머리 속 깊이 넣어주세요. ServiceHost 클래스의 생성자는 두 개의 파라미터를 받습니다.  첫 번째 파라미터는 호스팅 하려는 클래스의 타입이며, 두번째 파라미터는 Base Uri 입니다. Base Uri는 말 그대로 서비스의 기본 주소(address)를 나타냅니다.

ServiceHost 객체를 생성한 후에는 이 개체에 **AddServiceEndpoint** 메소드를 이용하여 endpoint를 추가하여 줍니다.

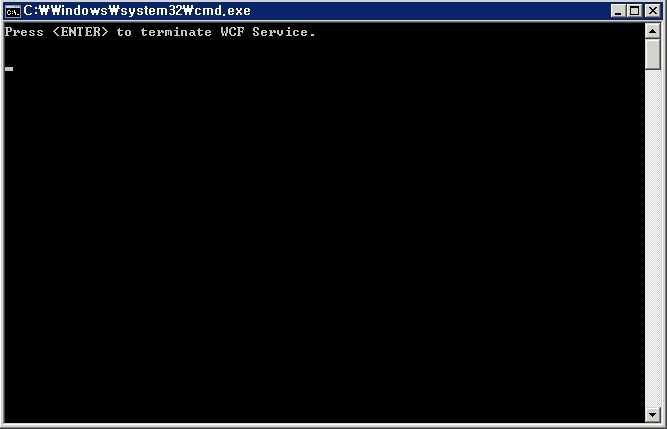
AddServiceEndpoint 메소드는 기본적으로 세 개의 파라미터를 필요로 하는데, 이 파라미터가 앞에서 설명했던 **Address, Binding, Contract** 입니다.

**Contract**는 서비스의 계약을 정의 했던 IProductService 인터페이스를 지정하구요, **Binding**은 **BasicHttpBinding** 클래스를 사용하였습니다. BasicHttpBinding 클래스는 WCF에서 기본적으로 제공하는 여러 Binding 개체 중에 하나로 이름에서 알 수 있듯이 기본적인 Http 프로토콜을 사용합니다. 그리고, 마지막 **Address** 는 빈 문자열을 주었습니다. 이렇게 빈 문자열을 주게 되면 endpoint의 주소는 ServiceHost 개체를 만들 때 사용했던 Base Uri와 동일한 값을 가지게 됩니다.

Endpoint 추가까지 끝이 나면 호스트를 수행할 준비가 모두 끝납니다. 이제 ServiceHost 개체의 **Open 메소드**를 사용하여 서비스를 시작할 수 있는데, Open 메소드는 각 채널에 맞는 리스너(listener)를 별도의 스레드로 수행을 하여 클라이언트의 요청을 처리할 수 있게 합니다.

서비스 구현은 모두 끝이 났습니다. 한번 실행해 볼께요~

실행을 하면 다음과 같은 Console 창이 뜹니다.

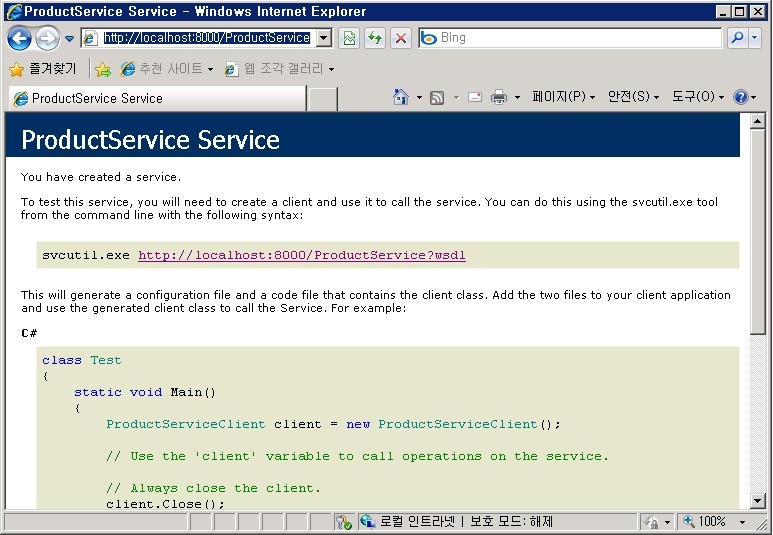
[](http://cfile22.uf.tistory.com/original/135CB51C4B45E5AF74580D)

이 창이 무사히 뜨면, 서비스가 아무 탈 없이 잘 동작하고 있다는 말입니다.

음,, 이것만 봐서는 믿음이 안 갈 수 있으니, 한번 검증해 보죠

이 콘솔 창을 그대로 띄워 놓은 상태에서 브라우저를 열고 주소창에 <http://localhost:8000/ProductService> (이 주소는 ServiceHost 개체를 생성할 때 입력했던 Base Uri) 를 입력해보겠습니다~

다음과 같은 내용을 확인할 수 있습니다. 짜잔~

[](http://cfile22.uf.tistory.com/original/117C8E1A4B45E6B5800991)