# XML 웹 서비스

## [ 웹 서비스 개론 ]

- 인터넷 환경에서 분산 응용 프로그램을 개발하는 가장 좋은 대안
  특정 회사의 독자적인 방법으로 제시된 개념이 아니라, 개방형 표준을 따르는 프로토콜과 XML을 가지고 분산 응용 프로그램 환경을 제공하고 있기 때문임

- 개발자가 개발한 다양한 응용 프로그램 로직들을 단순히 웹 서버에 올리는 것만으로 쉽게 웹 서비스를 개발할 수 있다.
  웹 서비스를 제공받는 클라이언트들은 개방형 표준 프로토콜인 HTTP, SOAP, HTML, XML을 사용해서 어디서나 웹 서비스를 제공받을 수 있다.

- 공부할 내용
  웹 서비스에 필요한 개방형 표준에 대한 이해
  웹 서비스를 이용한 응용 프로그램 구현

## [ 응용 프로그램 개발의 변화 ]

- 예전 웹 프로그래밍 : HTML 태그를 이용한 정적인 웹 페이지 구성
  디자인에 가까운 개념

- CGI or ASP를 이용한 동적인 페이지 생성이 가능해짐

- 분산 환경(TCP/IP, DCOM or CORBA등을 이용한 처리)으로 발전되어 짐

- XML웹 서비스 : 분산환경에서의 XML or SOAP를 통한 표준을 주도

## [ 웹 서비스 ]

- 표준 인터넷 프로토콜을 사용해서 접근이 가능한 응용 프로그램 로직임

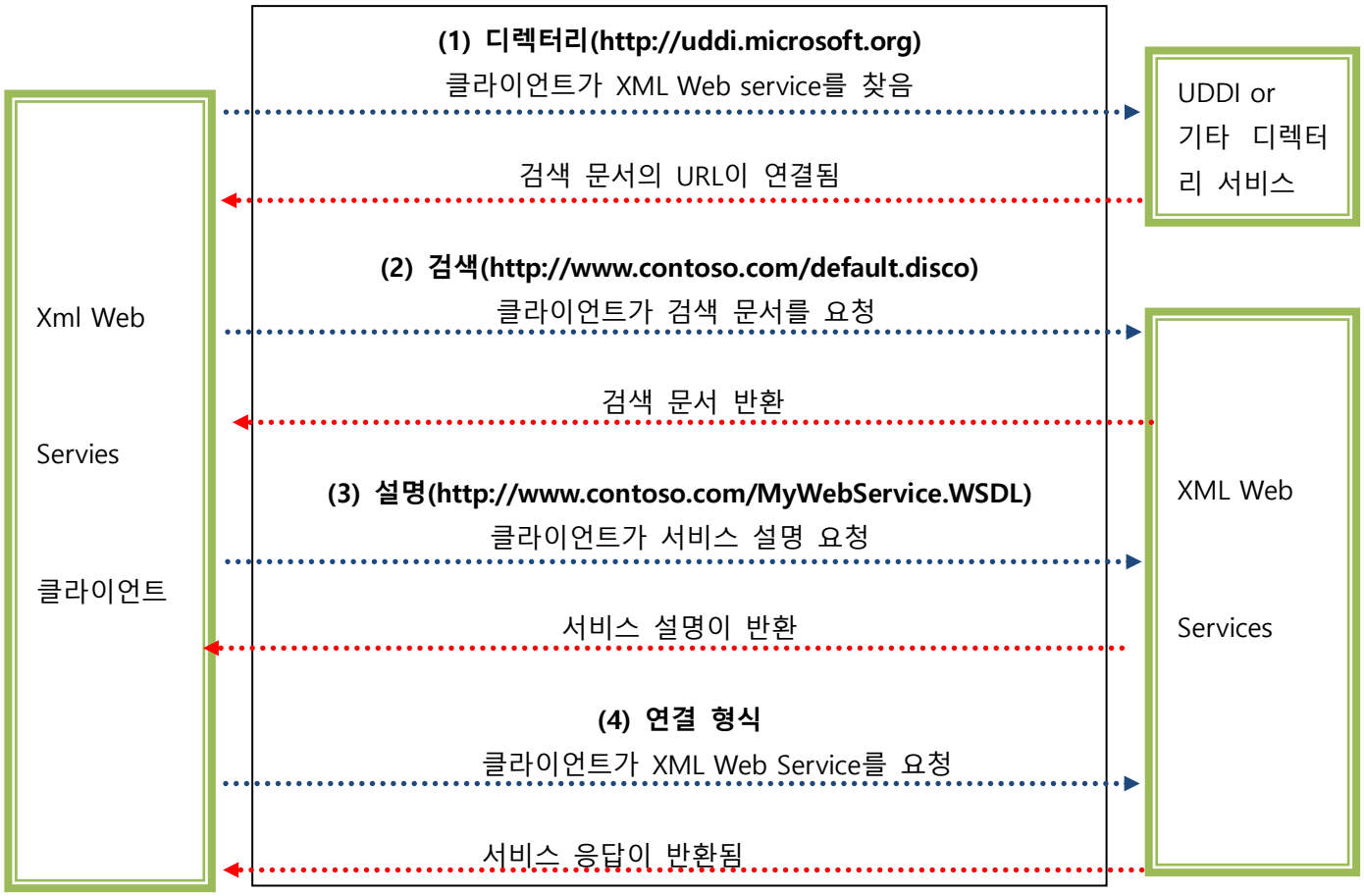
- 웹 서버 = 웹 그자체 + 컴포넌트 기반 개발의 장점

- 재사용이 가능한 컴포넌트 웹이라는 환경에서 접근하고 사용할 수 있슴

- 웹 서비스 환경을 실현하기 위한 구성 요소
  1) XML : 데이터를 표현하는 표준화된 방법
  2) SOAP : 일반적이고 확장 가능한 메시지 형식
  3) WSDL : 일반적이고 확장가능한 서비스 정의 언어[매뉴얼 – 리턴타입, 함수명, 매개변수리스트]
  4) Disco : 특정 웹 사이트에 위치한 서비스들을 찾는 방법
  5) UDDI : 서비스 제공자들을 찾는 방법

*) 위의 구성요소들은 모두 XML(eXtensible Markup Languge)을 사용해서 구현됨
*) 웹 서비스에서는 이 XML을 이용해서 데이터를 표현함

**웹 서비스의 흐름**




*)SOAP( Simple Access Transfer Protocol)

웹 서비스를 위한 프로토콜
표준화된 XML 기반의 메시징 프로토콜
SOAP 메시지, 데이터 인코딩을 위한 표준 및 요청/응답 처리 방법
SOAP메시지를 HTTP에 바인딩하는 필수적인 형식 제공

SOAP 프로토콜은 매우 복잡한 형태의 XML로 이루어짐
하지만, ASP.NET은 웹 서비스 개발에 SOAP 메시징에 대한 것을 자동으로 처리해줌

*) WSDL

웹 서비스에서 제공되는 컴포넌트를 설명하는 XML 문서
DCOM이나 CORBA처럼 독작적인 IDL을 사용하지 않고,
XML을 이용해서 객체의 이름, 메서드, 데이터 타입등을 개발자에게 알려줌

ASP.NET을 이용하면 자동으로 자신이 만든 컴포넌트에 대한 Web Services Description Language를 생성할 수 있슴

*) DISCO

특정 웹 사이트에서 서비스하는 웹 서비스를 검색하는 알고리즘을 정의해 놓은 것
웹 서비스의 설명이 들어있는 WSDL의 위치를 찾기 위함임

*) UDDI( Universal Description, Discovery, Integration)

다양한 웹 서비스들을 찾기 위한 디렉토리 역할
웹 사이트를 찾기 위한 검색 엔진에 비유됨
즉, 전 세계의 수많은 웹 사이트들이 제공하는 웹 서비스들을 디렉터리 형태로 관리해줌

**[ 사례를 통해 알아보는 웹 서비스 ]**

"가격 정보 사이트" 개발

- 모든 물품에 대한 가격을 알아야 하고, 거래를 맺게 해주어야 함

1) 전자상거래를 하는 모든 기업의 웹 사이트를 돌아다니며 정보를 얻거나, **전화나 팩스로** 정보를 얻음
수집한 정보를 모두 자기의 사이트에 맞게 편집
고객의 신용정보는 가입 시에 얻은 정보를 이용해서 신용 확인할 수 있는 곳에 문의
자신의 웹 사이트를 갱신

2) 전자상거래를 하는 모든 기업의 웹 사이트와 파트너 관계를 맺음
자기와 **정보를 교환할 수 있는 규칙을 정해서 서로 통신(독자적인 프로토콜 사용)**
고객의 신용 정보는 신용 정보만 서비스하는 곳에 의뢰해서 웹에서 알 수 있도록 처리
이렇게 자동화 하여 자신의 가격 정보 사이트를 실시간으로 유지
=> 해당 기업의 하드웨어 시스템이 다르면 불가능함

3) 웹 서비스 이용 방식
가격 정보 사이트에 서비스를 제공하는 또 다른 웹 서버가 존재
웹 서버는 user가 아닌 또 다른 기업에 서비스를 해줌

서로 간의 정보교환은 HTTP를 통해 **XML로 교환함으로** 시스템의 투명성, 데이터의 투명성을 보장 : SOAP의 개념임

클라이언트는 웹 브라우저가 아닌 GUI응용 프로그램, PDA, 휴대폰등 모든 기기로 확대가 아주 쉽게 이루어짐

**[ 개발자의 입장에서 바라본 웹 서비스 ]**

- 분산 프로그래밍을 개발하는 개발자의 관점에서 웹 서비스는 원격지에 있는 객체를 호출해서 그 객체가 가지고 있는 기능 이용

  웹 서비스 또한 목적은 동일  하지만 방법의 차이가 존재

  - 훨씬 개방적이고 구현이 용이
  - 통신 프로토콜 : 객체의 요청, 메서드 호출, 리턴값 반환등을 SOAP(HTTP와 XML을 이용해서 객체에 접근하는 규칙)을 이용해서 표준화함

  - 객체를 노출시키는 방법 : 독자적인 IDL을 사용하지 않고, XML을 이용해서 객체의 이름, 메서드, 데이터 타입등을 사용자에게 알려주는 WSDL을 이용

  *) 컴포넌트를 개발해서 팔때 : 업데이트 되면 배포된 컴포넌트 모두 수정
  웹 서비스는 자기가 서비스할 객체를 자신의 웹 서버에 노출시킴으로  해결 가능

**[ ASP.NET에서 바라본 웹 서비스 ]**

ASP.NET은 XML 웹 서비스를 보다 쉽게 개발할 수 있도록 하부 구조 제공
따라서 단지 제공하고자 하는 서비스에 대한 구현에만 몰두 할 수 있다.

개발 단계

1) .asmx 확장자를 가진 파일 생성
2) 생성한 파일에 지시어를 사용해서 XML 웹 서비스를 선언
3) 생성한 파일에 개발하려는 XML 웹 서비스가 서비스하는 메서드들을 정의

**[ ASP.NET에서 바라본 웹 서비스의 동작 과정 ]**

1) 클라이언트가 수행되는 시점에서 웹 프록시 객체를 생성하고, 클라이언트는 이 웹 프록시 객체가 가지고 있는 메서드를 호출

2) 웹 프록시는 클라이언트에 투명하게 매서드 호출을 HTTP와 XML을 이용해서 마샬링하고 인터넷으로 보냄

3) ASP.NET은 XML 형식의 HTTP 요청을 받고, XML 안의 메서드 이름과 인자를 알아내서 해당 .asmx 파일에 명시된 객체를 생성 그리고 객체내의 해당 메서드를 호출

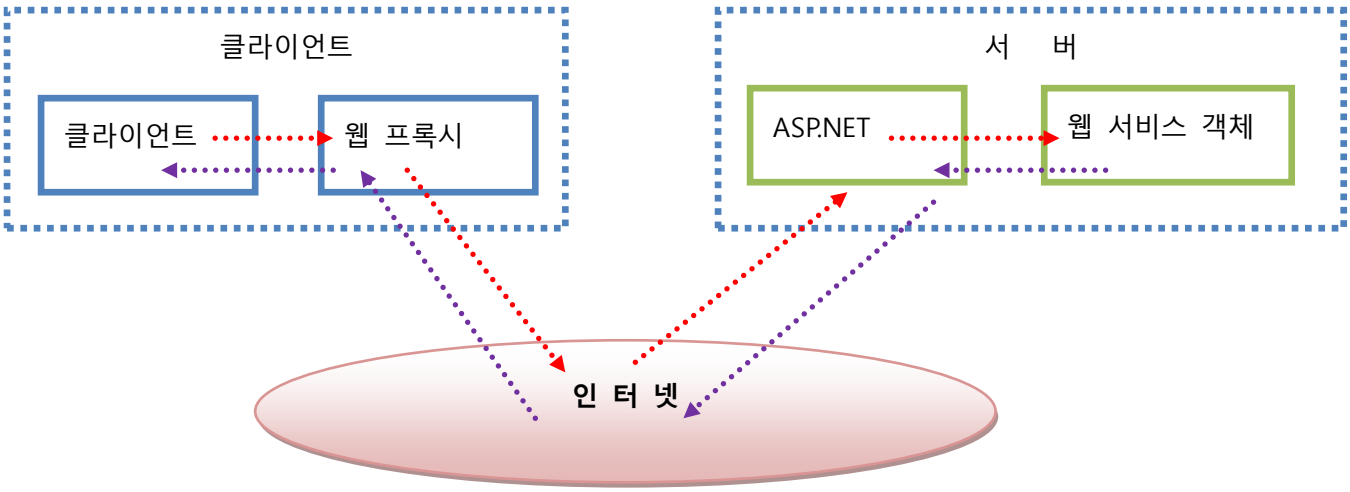
4) 웹 서비스 객체는 ASP.NET 에 결과를 리턴

5) ASP.NET은 웹 서비스 객체로부터 리턴 받은 값을 인터넷 상에 있는 클라이언트에게 보내기 위해서, 리턴 값을 XML형식으로 바꾸고 HTTP를 통해서 보냄

6) 웹 프록시는 HTTP를 통해서 XML 형식으로 결과 값을 받고, 클라이언트가 이해할 수 있는 요청한 메서드의 리턴값으로 변경

클라이언트는 마치 로컬에 있는 객체의 메서드를 호출해서 리턴값을 얻은 것과 동일하게 리턴값을 처리

*) 웹 서비스가 존재하는 웹 사이트를 찾아주는 UDDI or
찾은 웹 사이트에서 제공되는 서비스를 찾는 DISCO는 동작 과정에서 생략



# 웹 서비스 구현

**[ 웹 서비스 만들기 1 ]**

ASP.NET으로 웹 서비스 개발시 , 복잡한 SOAP메시징과 HTTP 전송등의 작업을 직접 할 필요 없슴
개발자는 단지 웹 서비스로 제공되는 클래스를 정의하기 위해서 필요한 메서드 등의 정의만 하면 됨

**1) 웹 서비스 선언**

- .asmx 파일 확장자를 가진 파일로 정의됨

- 실제 코드는 .asmx 파일이나 미리 컴파일된 클래스(.dll파일)로 되어 있슴

- .asmx 파일에 WebService 지시어를 추가해서 웹 서비스를 선언함
  <%@ WebService Language="C#" Class="클래스 이름" %>

**2) 웹 서비스 클래스 만들기**

- .asmx 파일로 작성

```
// Hello.asmx

    // 프로그래밍 언어와 클래스 이름에 대한 선언
    <%@ WebService Language="C#" Class="ShowHelloService" %>
    using System;
    using System.Web.Services;

    // WebService 클래스에서 파생시켜 사용자 클래스 정의
    public class ShowHelloService : WebService
    {
        [WebMethod]     // 웹 메서드 속성을 가진다는 것을 명시
        public string ShowHello()
        {
            return "Hello WebService!!";
        }
    }
```

**3) 웹 폴더에 저장하고 실행**

- 테스트 초기 화면
- 메서드 호출화면

  => 작성한 문자열이 XML 형식의 값으로 표시되는 것 확인 가능

⇨ 개발한 부분

"Hello WebService"문자열을 주는 웹 메서드를 가진 ShowHelloService 클래스 정의
.asmx 파일로 저장
적당한 웹 폴더에 넣어둠
클라이언트가 웹 서비스를 호출해서 리턴 값을 XML 형태로 얻은 것임

⇨ 개발하지 않은 부분

클라이언트가 ShowHelloService 웹 서비스를 어떤 웹 사이트에서 제공하는지를 알려주지 않음(UDDI)
이 웹서비스를 제공하는 웹 사이트의 어떤 위치에 있는지 알려주지 않음(DISCO)

⇨ 자동으로 작성된 작업

이 웹 서비스가 제공하는 웹 서비스에 대한 XML 형식의 설명 문서 자동 생성(WSDL)
실제 클라이언트가 메서드 호출하는 메시징 형식과 이 웹 서비스를 제공하는 서버가 리턴 값을 돌려줄 때의 메시징 형식 직접 구현 필요 없슴(SOAP)

*) WSDL : 브라우저 화면 상단의 "서비스 설명" 클릭

- XML 형식으로 변환되어 있슴
- 클라이언트가 ShowHello() 라는 이름의 메서드를 호출하려면, 그 메서드가 인자를 갖지 않고, string 타입을 리턴한다는 것을 알려주는 역할을 수행
- ASP.NET에서 자동 수행

*)SOAP :

- ShowHello() 웹 메서드를 호출한 화면의 아래 부분에 나와있슴
- XML로 구현되어 있고 SOAP 요청과 응답에 대한 정보를 보여주게 됨
- SOAP은 HTTP 와 같은 일반 전송 프로토콜에 표준 데이터 형식인 XML을 이용해 데이터를 교환하는 프로토콜임
  SOAP은 단지 문자열로 마샬링된 XML로써 POST를 통해 HTTP 메시지의 본문으로 전송되는 데이터임

**[ 웹 서비스 만들기 2 ]**

코드-비하인드를 사용해서 ShowHelloServie 서비스 만들기

- 웹 서비스 개발은 UI가 없으므로, .asmx파일 수정을 통해 쉽게 할 수 있슴

```
// Hello.asmx
    <%@ WebService Language="C#" Codebehind="Hello.asmx.cs"
        Class="ShowHelloService" %>
```

```
// Hello.asmx.cs
    using System;
    using System.Web.Services;

    public class ShowHelloService : WebService
    {
        [WebMethod] // 웹 메서드 속성을 가진다는 것을 명시
        public string ShowHello()
        {
            return "Hello WebService!!";
        }
    }
```

1) 정의된 .cs 파일을 .dll 파일로 컴파일헤서 어셈블리로 만듬

csc /t:library Hello.asmx.cs
=> c://Documents and Settings₩Administrator₩Hello.asmx.dll

2) .asmx 파일이 있는 위치에 bin 폴더를 생성해서 저장

3) Web Browser에서 접근

*) 코드 -비하인드를 사용하면 소스 코드가 어셈블리로 저장되므로
- 코드가 유출될 우려가 없슴
- 이미 컴파일되어 있으므로 초기 요청에 빠르게 동작
- 다른 응용 프로그램에서 재사용 가능

*) ShowHelloServie 웹 서비스 알리기
다른 개발자에게 알리는 방법 : 이것은 UDDI가 맡고 있슴
UDDI는 다시 말해 웹 서비스들을 등록하고, 다른 사용자들이 웹 서비스를 검색할 수 있게 해줌
현재 MS(http://uddi.microsoft.com), 아리바(http://www.ariba.com), IBM(http://www-3.ibm.com/service/uddi)
등이 이러한 것을 제공하고 있으며, 서로의 정보를 공유하고 있슴

*) 참고자료 : http://www.uddi.org

# 클라이언트를 위한 웹 프록시 구현

**[ 클라이언트가 사용할 수 있는 웹 Proxy 만들기 ]**

클라이언트가 ShowHelloService 웹 서비스가 어떤 사이트에 위치하고 있다는 것을 안다고 가정
웹 프록시는 클라이언트가 원격지에서 웹 서비스로 제공되는 객체를 마치 자신의 컴퓨터의 객체를 사용하는 것처럼 클라이언트를 구현하는 개발자에게 투명성을 제공해 주는 역할을 함
즉, 클라이언트는 로컬의 웹 프록시에 정의되어 있는 클래스에 대한 객체를 생성하고 객체의 메서드를 호출해서 사용함

- 웹 프록시는 클라이언트가 호출한 메서드가 실제 정의되어 있는 원격지로 보내게 됨
- 이 원격지는 웹 서비스를 제공하고 있는 곳을 의미
- 이제 웹 서비스에서 호출이 발생되고 수행된 후에 리턴값을 다시 가져옴
- 그리고 웹 프록시는 클라이언트에게 마치 로컬에 있는 메서드처럼 리턴 값을 넘겨줌

**1) 웹 프록시 생성**

- 웹 프록시는 닷넷에서 제공하는 언어를 가지고 직접 생성 가능
  .NET에서는 명령어 한 줄이면 자동으로 만들어짐
  WSDL.exe : 웹 프록시를 생성시키는 명령어

```
WSDL /o:ShowHelloServiceProxy.cs http://localhost/CCM/Hello.asmx?WSDL
```

ShowHelloServiceProxy.cs 파일로 생성됨

- 생성된 파일을 dll 형태로 컴파일해야 함

```
csc /t:library ShowHelloServiceProxty.cs
```

ShowHelloServiceProxy.dll이 생성됨

# ShowHelloService 웹 서비스를 제공받는 클라이언트

**[ 웹 클라이언트 구현 방식 ]**

웹 서비스를 제공받는 클라이언트는 크게 세 가지 형태로 나눌 수 있슴

- 웹 응용 프로그램
- GUI 응용 프로그램
- 콘솔 응용 프로그램

**[ ASP.NET 웹 응용 프로그램 형태의 클라이언트 구현 ]**

- ShowHelloService 웹 서비스에 대한 웹 프록시를 생성하면, 클라이언트의 개발은 웹 프록시를 참조해서 구현하면 됨
- 웹 프록시는 클라이언트 파일인 .aspx파일과 동일 폴더에 둔다.

```
// HelloClient.aspx ---------------------------------------------
<%-- HelloClient.aspx --%>
<%@ Page Language="c#" %>
<%@ Import Namespace="System" %>
<script runat="server">
	void myButton_Click(Object sender, EventArgs e)
	{
		// 웹 서비스 클래스에 대한 객체를 생성한다.
		ShowHelloService shs = new ShowHelloService();

		// 웹 서비스 메서드를 호출한다.
		myTextBox.Text = shs.ShowHello();
	}
</script>
<html>
<body>
<form id="Form1" method="post" runat="server">
	<asp:TextBox id="myTextBox" runat = "server">
	</asp:TextBox>
	<asp:Button id="myButton"
		Text="웹서비스요청"
		runat="server"
		BorderStyle="Solid"
		OnClick="myButton_Click"/>
</form>
</body>
</html>
```

➔ 웹 프록시는 클라이언트의 호출을 자동으로 SOAP 메시지 형태로 변환해서 HTTP 프로토콜을 통해 웹 서비스가 위치한 곳으로 메서드와 인자를 넘김
원격지에서 SOAP 메시지를 통해 얻은 리턴 값은 다시 ASP.NET웹 클라이언트가 이해할 수 있는 리턴값으로 돌려줌

# ShowHelloService 웹 서비스를 제공받는 클라이언트

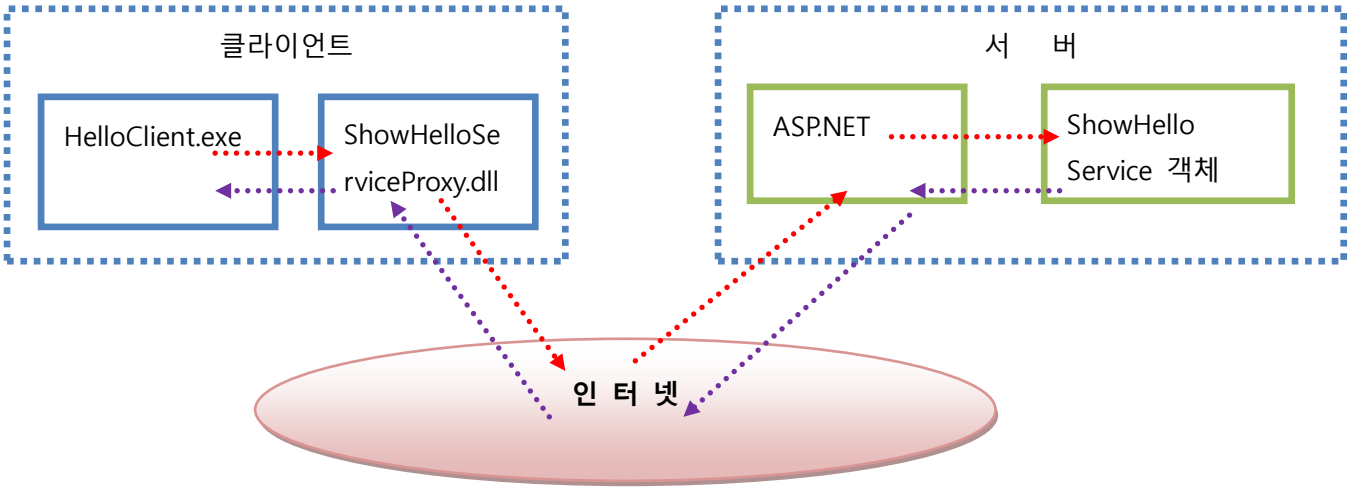
**[ 윈도우 폼 응용 프로그램 형태의 클라이언트 구현 ]**

- ASP.NET웹 응용 프로그램과 동일한 웹 프록시를 사용하며, ShowHelloServiceProxy,dll 파일을 참조함

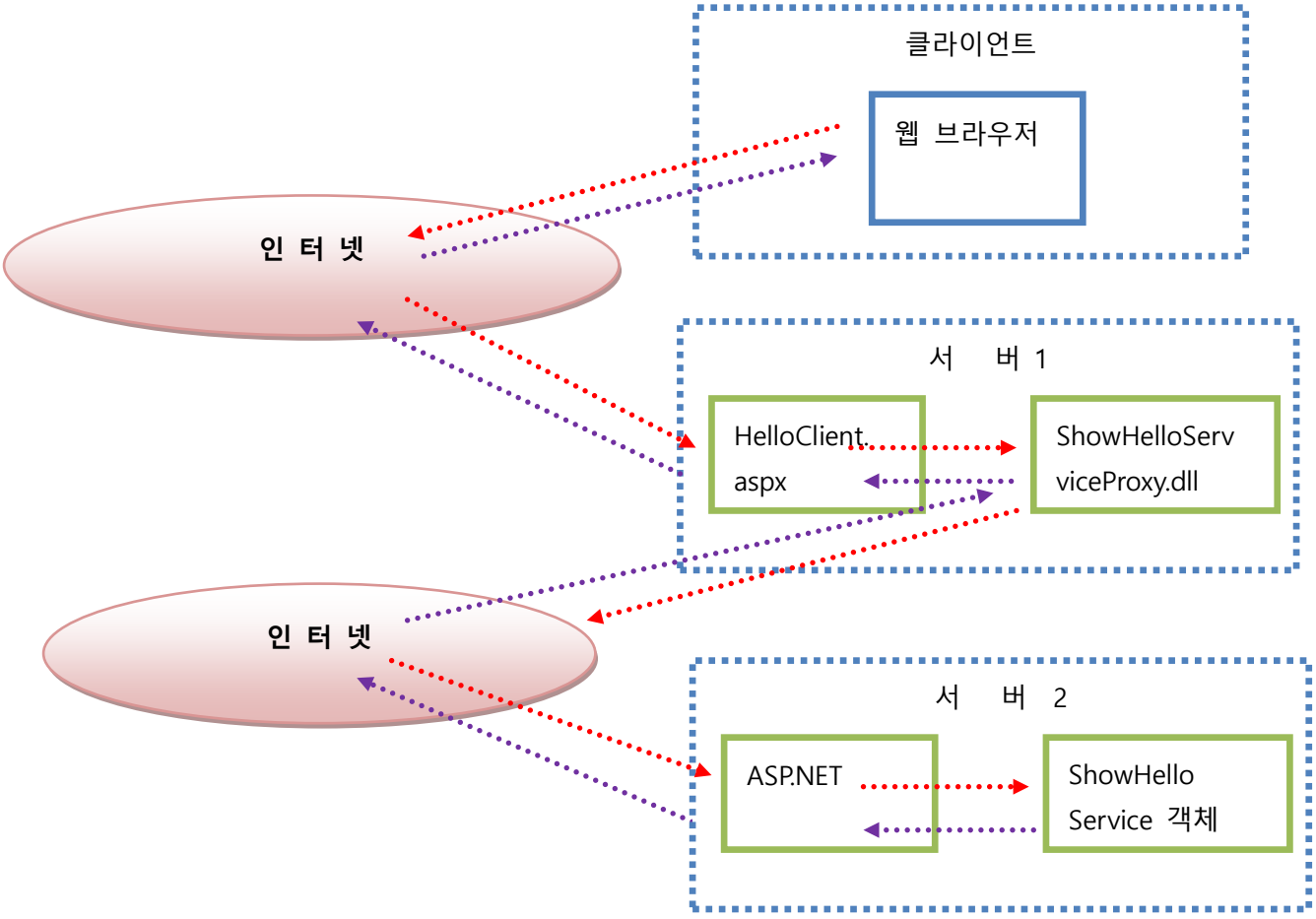
```
// HelloClient.cs  ------------------------------------------

using System;              using System.Drawing;              using System.Windows.Forms;

class HelloClient : Form
{
        Button btn;                TextBox tb;

        HelloClient()
        {
                Text = "HelloClient";
                btn = new Button();
                btn.Location = new Point(180, 110);
                btn.Size = new Size(100, 20);
                btn.Text = "웹 서비스요청";
                btn.Click += new System.EventHandler(btn_Click);
                Controls.Add(btn);

                tb = new TextBox();
                tb.Location = new Point(10, 110);
                tb.Size = new Size(150, 20);
                Controls.Add(tb);
        }
        static void Main()
        {
                Application.Run(new HelloClient());
        }
        protected void btn_Click(Object o, EventArgs e)
        {
                // 웹 서비스 클래스에 대한 객체를 생성한다.
                ShowHelloService shs = new ShowHelloService();

                // 웹 서비스 메서드를 호출한다.
                tb.Text = shs.ShowHello();
        }
}
```

csc /r:ShowHelloServiceProxy.dll HelloClient.cs 로 컴파일.....
만들어진 HelloClient.exe 실행 파일로 실행.

[ ShowHelloService 웹 서비스를 이용한 윈도우 클라이언트의 전체적인 동작 과정 ]




[ ShowHelloService 웹 서비스를 이용한 ASP.NET 클라이언트의 전체적인 동작 과정 ]



# 웹 서비스 구현 : VS.NET을 이용한 프로그래밍

## [ 이미지 파일을 제공하는 웹 서비스 만들기 ]

3개의 웹 메서드

GetPicture() : 해당 파일 이름의 이미지 파일을 바이트로 리턴

GetPictureList() : 현재 이 웹 서비스가 제공하는 이미지 파일들의 목록을 문자열 배열로 리턴

UploadPicture() : 인자로 받은 파일 이름으로 인자로 받은 바이트 배열을 서버에 저장
성공하면 true, 실패하면 false를 리턴

## [ STEP1 : ASP.NET 웹 서비스 탬플릿 선택 ]

위치 입력 : HTTP => http://localhost/PictureService

*) 생성된 파일은 코드-비하인드 기법으로 구성됨
웹 서비스는 대부분 UI부분이 없으므로, cs파일을 직접 작성하게 됨
자동으로 생성된 Service1.asmx.cs 파일을 직접 변경

## [ STEP2 : 코드 구성 ]

```
// Service.asmx   ----------------------------------------

<%@ WebService Language="c#" Codebehind="Service1.asmx.cs"
                                                    Class="PictureService.Service1" %>
```

```
        //Service.cs ---------------------------------------------

        using System;
        using System.Collections;
        using System.ComponentModel;
        using System.Data;
        using System.Diagnostics;
        using System.Web;
        using System.Web.Services;

        using System.IO;              // 파일입출력과 관련된 맴버 사용

// 이어서...
```

```
namespace PictureService
{
    /// <summary>
    /// Service1에 대한 요약 설명입니다.
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            //CODEGEN: 이 호출은 ASP.NET 웹 서비스 디자이너에 필요합니다.
            InitializeComponent();
        }

        #region Component Designer generated code

        //웹 서비스 디자이너에 필요합니다.
        private IContainer components = null;

        /// <summary>
        /// 디자이너 지원에 필요한 메서드입니다.
        /// 이 메서드의 내용을 코드 편집기로 수정하지 마십시오.
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
/// 사용 중인 모든 리소스를 정리합니다.
        /// </summary>
        protected override void Dispose(bool disposing)
        {
            if (disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
#endregion
```

```
// 해당 이미지 파일을 서비스해주는 메서드
[WebMethod]    // 웹메소드 속성을 가진다는 것을 명시
public byte[] GetPicture(string strFileName)
{
        byte[] bytePic = { 0 };          // 바이트 배열을 하나 만든다.
        try
        {
             // 해당 이미지 파일을 스트림 형식으로 오픈한다.
            FileStream picFileStream = new
           FileStream(@"C:₩Inetpub₩wwwroot₩PictureService₩Pics₩" + strFileName, FileMode.Open,
           FileAccess.Read, FileShare.Read);

           // 이미지 파일 스트림을 읽을 객체를 하나 만든다.
           BinaryReader picReader = new BinaryReader(picFileStream);
           // 이미지 파일을 바이트 배열에 넣는다.
           bytePic = picReader.ReadBytes(Convert.ToInt32(picFileStream.Length));
           // 파일스트림을 닫는다.
           picFileStream.Close();
           // 이미지 파일이 들어있는 바이트 배열을 리턴한다.
           return bytePic;
        }
        catch
        {
            // 초기값을 그냥 리턴한다.
            return bytePic;
        }
}

// 이미지 파일 목록을 리턴해주는 메서드
[WebMethod]
public string[] GetPictureList()
{
        // 이미지 파일이 들어있는 디렉토리에서 파일 이름들을 문자열 배열에 넣는다.
        string[] strPicList = Directory.GetFiles(@"C:₩Inetpub₩wwwroot₩PictureService₩Pics₩");

        // 파일 경로를 뺀 파일 이름만 다시 추출한다.
        for (int i = 0; i < strPicList.Length; i++)
        {
               FileInfo fi = new FileInfo(strPicList[i]);
               strPicList[i] = fi.Name;
         }
         // 이 이미지 파일 이름들을 리턴한다.
        return strPicList;
}
```

```
// 클라이언트들이 업로드하는 이미지 파일들을 저장하는 메서드
[WebMethod]
public bool UploadPicture(string strFileName, byte[] bytePic)
{
        try
        {
            // 주어진 이미지 파일의 이름으로 파일을 하나 만든다.
           FileStream writeFileStream = new FileStream(
           @"C:₩Inetpub₩wwwroot₩PictureService₩Pics₩" + strFileName, FileMode.Create,
           FileAccess.Write);

           // 이 파일에 바이너리를 넣기 위해 BinaryWriter 객체 생성
           BinaryWriter picWriter = new BinaryWriter(writeFileStream);
           // 바이트 배열로 받은 이미지를 파일에 쓴다.
           picWriter.Write(bytePic);
           // 파일스트림을 닫는다.
           writeFileStream.Close();
           // 업로드 성공
           return true;
        }
        catch (Exception e)
        {
            // 업로드 실패
           return false;
        }
}
}
}
```

**[ STEP3 : 컴파일 하기 전에 코드에서 사용한 "Pics"폴더를 생성 ]**

- 임의의 이미지 파일들 저장

**[ STEP4 : 해당 메서드들 확인 ]**

# 웹 서비를 제공받는 윈폼 클라이언트 구현 : VS.NET 사용

## [ STEP 1 : Windows응용 프로그램 폼 생성 및 디자인 ]

- PictureClient 프로젝트 생성

- 2개의 폼 구현

(1) PictureClient의 디자인

| 컨트롤 | 속성 변경 |
|---|---|
| **Form** | Name = "Form1"<br>Text = "PictureClient"<br>MaximizeBox = False<br>Menu = mainMenu1<br>FormBorderStyle = FixedSingle |
| **MainMenu** | Name = mainMenu1 |
| **MenuItem** | Name = menuItem1<br>Text = "그림 목록 보기" |
| **MenuItem** | Name = menuItem2<br>Text = "그림 업로드 하기" |

(2) PicListForm

| 컨트롤 | 속성 변경 |
|---|---|
| **Listbox** | |
| | Name = mainMenu1 |
| **Button** | Text = "선택한 이미지 보기" |

⇨ 소스 복사

## [ STEP 2 : PictureService 웹 서비스 제공받기 ]

웹 서비스에서 제공받는 것도 네임스페이스에 있는 객체들임

외부 객체 참조 방법

[솔루션 탐색기 ] [참조]=>[웹 참조] : localhost를 코드에서 알기쉽게 PictureService로 변경

WSDL같은 명령을 할 필요 없음 (자동으로 생성시켜 줌)

우리가 작성한 Service는 UDDI등록이 되어 있지 않으므로, URL을 통해 찾아온다.

=> 솔루션 탐색기에 생성된 Service 목록 확인

웹 서비스 테스트 화면이 왼쪽창에 오른쪽 창에는 "계약보기", "설명서 보기"링크가 나타남

- 계약 보기 : PictureService웹 서비스의 WSDL을 출력해줌
- 설명서 보기 : 테스트 화면을 보여줌

**[ 전체 적인 동작과정 ]**

- 웹 프록시에 해당되는 Reference.cs 파일은 명령 프롬프트에서 WSDL.exe 유틸리티를 사용해서 직접 생성시킨 것이 아니라 웹 참조 추가를 통해 자동으로 생성됨
- Reference.cs파일은 웹 참조 폴더에 생성됨
- 해당 파일이 cs파일이던 dll파일이던 상관없슴
-