**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 ]**