서해원

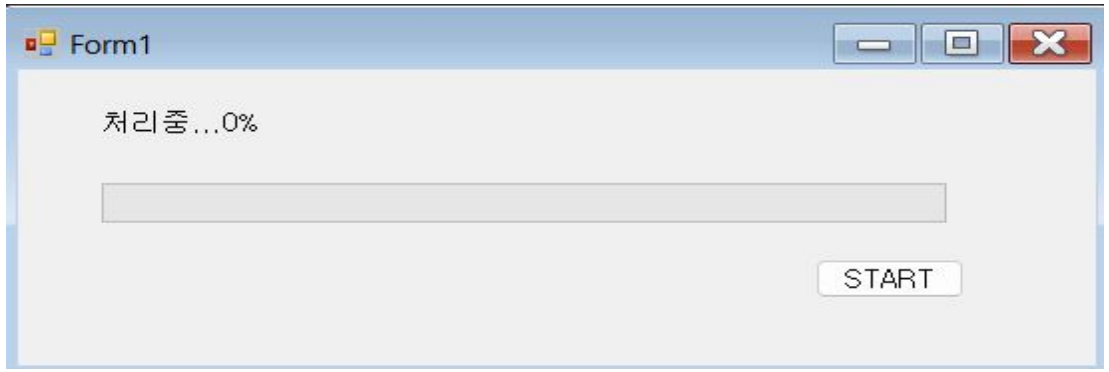
C# 프로젝트
Text to Speech

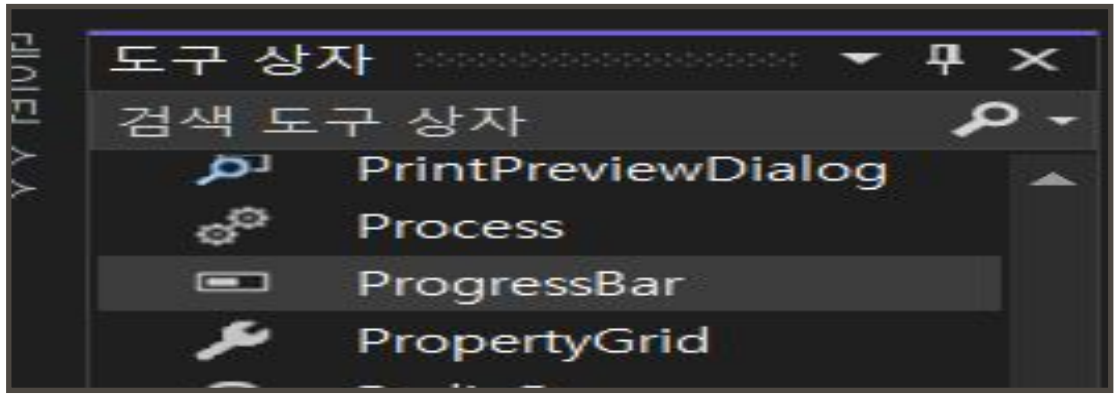
목차

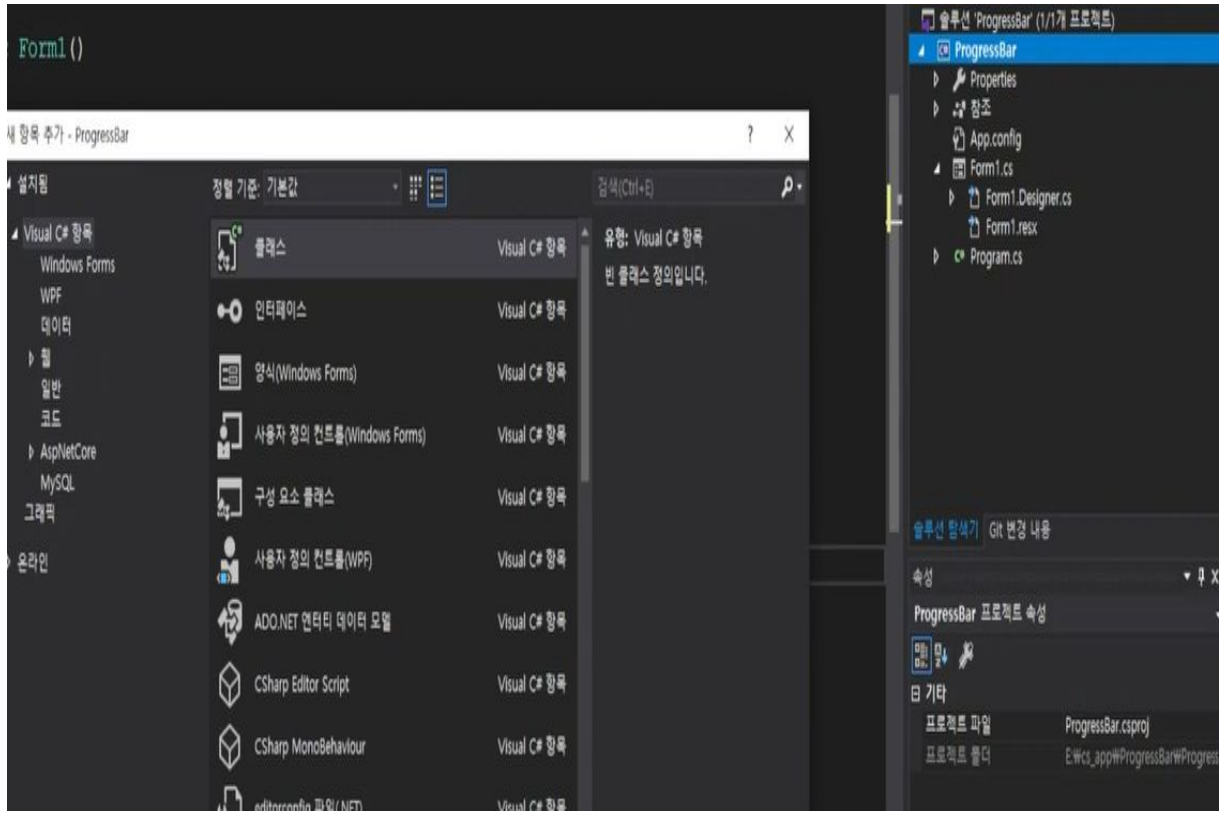
1-1. PROGRESS-BAR

이미지, 도구상자

클래스 추가_새항목추가







1-2. PROGRESS-BAR

클래스 추가_코딩

```
namespace PROJECT_csharp_01
{
    참조 3개
    internal class Progress
    {
        참조 3개
        public int PercentComplete { get; set; }
    }
}
```

추가 설명

자동 구현 프로퍼티(Auto-Implemented Property)

은닉성을 위해서는
필요한 부분만 getter, setter를 구현해서 사용함.

한 변수에 getter와 setter까지 하면 변수가 많아질수록
이 수 또한 엄청 많아지게 됨

C#에서는 이러한 것들을
한 뭉치로 묶을 수 있는 프로퍼티를 제공

1. get 접근자: 필드로부터 값을 읽어옴
2. set 접근자: 필드에 값을 할당

1-3. PROGRESS-BAR

```
private async void btnSTART_Click(object sender, EventArgs e)
{
    List<string> list = new List<string>();
    for (int i = 0; i < 100; i++)
    {
        list.Add(i.ToString());
    }//for

    label1.Text = "작업중.....";

    var progress = new Progress<Progress>();
    progress.ProgressChanged += (o, report) =>
    {
        label1.Text = string.Format("처리중...{0}%", report.PercentComplete);

        progressBar1.Value = report.PercentComplete;

        progressBar1.Update();
    };//
    await ProcessData(list, progress);

    label1.Text = "완료!!";

    Form2 f2 = new Form2();
    f2.Show();
}// click
```

async & await
: 비동기식 코드를 동기식으로 표현하여 간단하게 나타내는 것

await
: await는 비동기 작업의 흐름을 제어하는 키워드라고 할 수 있다.
나아가 비동기 작업이 실행될 수 있는 곳이 바로 await이다.
await 키워드는 반드시 async함수 안에서만 사용할 수 있다,

async
: async 키워드는 해당 메서드 내에 await 키워드를 사용할 수 있게 만들어준다

<!>async/await는 기본적으로 Task를 사용해야 하는데, 리턴타입이 없으면 Task 객체를, 리턴타입이 있으면 Task<T> 객체를 사용합니다.
(UI 이벤트핸들러와 같이 특별한 케이스에는 void를 사용하지만, 예외적인 상황으로 이해.)

ProgressChanged 이벤트
: 진척 사항을 전달 처리하는 ProgressChanged 이벤트

ProgressBar.Value 속성
:진행률 표시줄의 현재 위치를 가져오거나 설정

Update() 메서드
: 무효화된 영역을 다시 그리게 한다.

1-4. PROGRESS-BAR

```
await ProcessData(list, progress);
```

```
private Task ProcessData(List<string> list, IProgress<Progress> progress)
{
    int index = 1;
    int totalProcess = list.Count;

    var progressReport = new Progress();

    return Task.Run(() =>
    {
        for (int i = 0; i < totalProcess; i++)
        {
            progressReport.PercentComplete = index++ * 100
          / totalProcess;
            progress.Report(progressReport);
            Thread.Sleep(80);
        }//for
    });//run
} // private Task ProcessData
```

Task-based Asynchronous Pattern(이하 TAP)
.Net framework 4.0에서 나온 개념으로 **Task** 에서 비동기를 실행.
TAP은 단일 메소드를 사용하여 비동기 작업의 시작과 완료를 나타냄

Task
: 클래스
Task의 작동방식은 .NET Framework에서 관리되고 있는 ThreadPool에서 작동합니다.
ThreadPool 에서 더 편리하게 사용하는 것
: 단발적이고 짧은 동작들을 수행하는 경우 사용한다.

Task.Run():
지정한 작업을 ThreadPool에서 실행하도록 큐에 대기시키고
해당 작업에 대한 작업 또는 Task<TResult> 핸들을 반환
Run(정적 메소드) 를 이용하여 객체를 만들지 않고
바로 쓰레드를 실행하는 것도 가능하다.

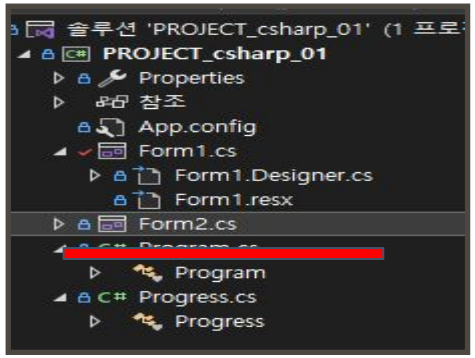
Thread.Sleep(80);
: 실행중인 스레드를 잠시 멈추게 하고 싶다면
Thread 클래스의 정적 메소드인 sleep() 메소드

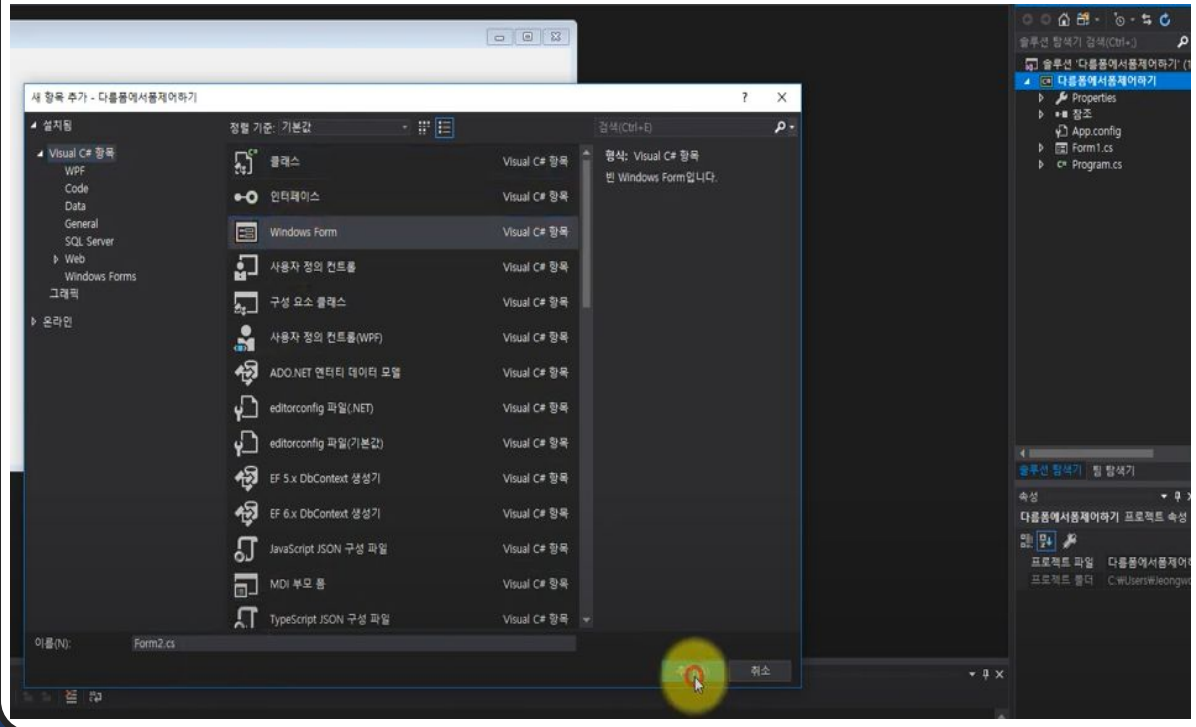
IProgrss<Progress> progress
IProgress 인터페이스를 통해 Progress를 전달받음
Progress객체가 IProgress를 상속받고 있기 때문에
Progress객체로 전달 받으면 됨.

Report() 메소드
: 진행률 업데이트를 보고함
: IProgress인터페이스의 Report함수에
인자로 원하는 값을 진행상황으로 넘겨주면 됨.

2-1. 새로운 창 띄우기

```
Form2 f2 = new Form2();
f2.Show();
```

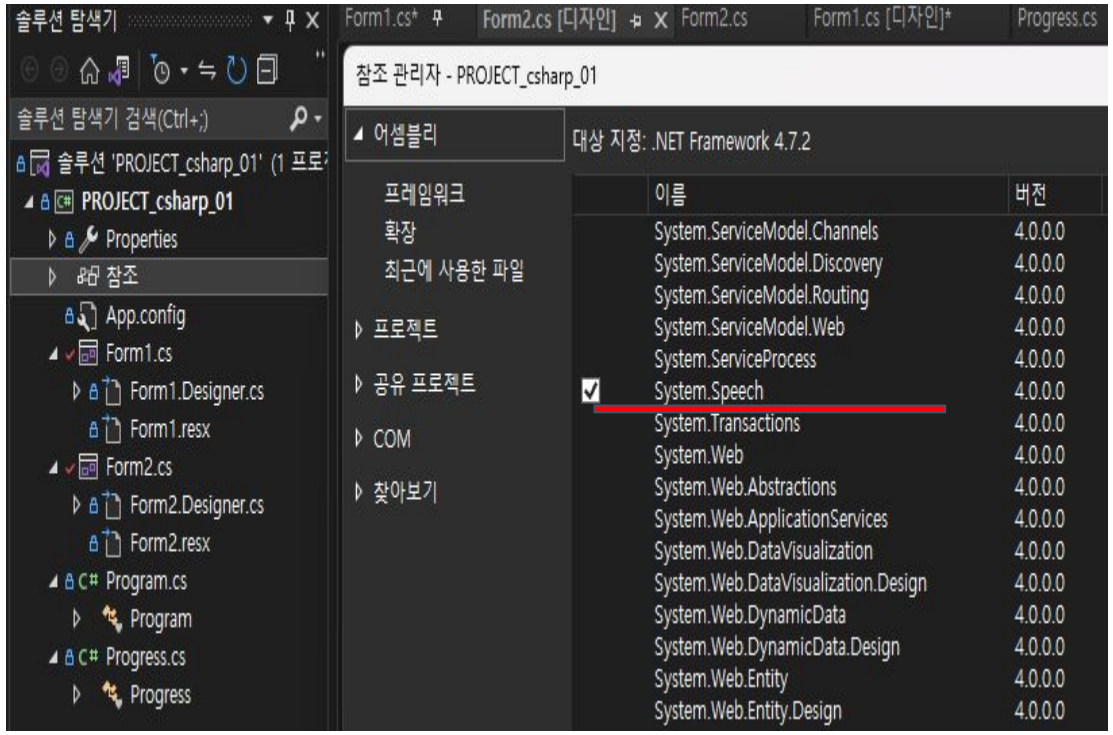




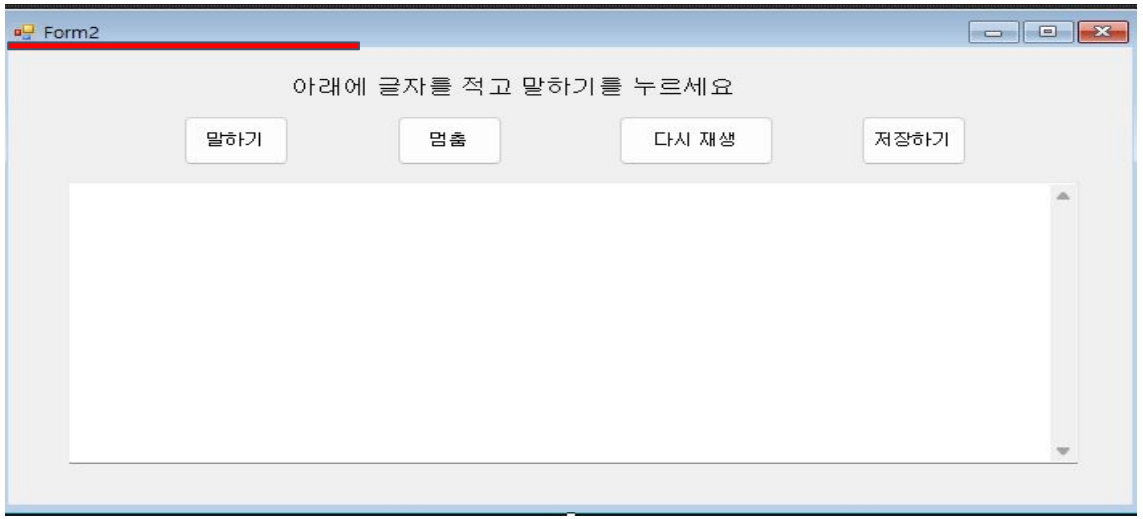
기존 폼에서 새 폼 열기

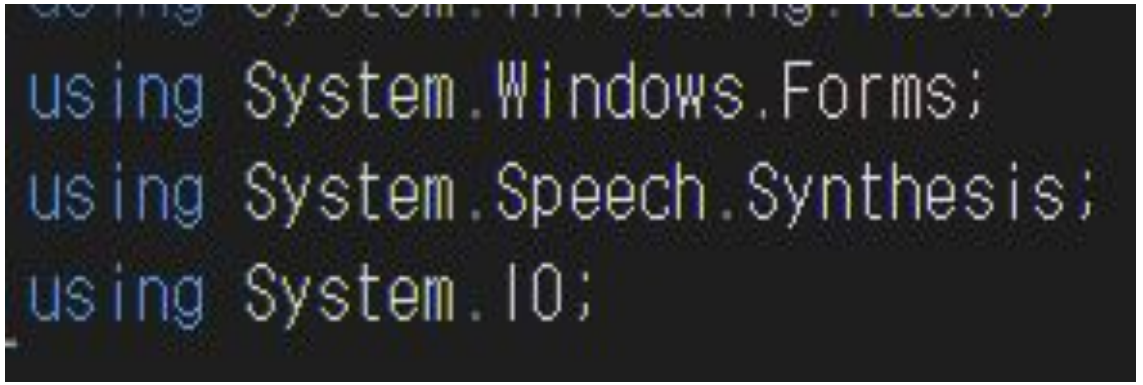
1. 새항목 추가-> **Windows.Form** 추가
2. 솔루션 탐색기 추가 확인
3. 진행을 나타내는 바가 다 끝난 후에 나타나도록 설정
4. **await** 다음에 나오도록 버튼 클릭이벤트의 맨마지막에 위치함
5. **f2** 는 새롭게 띄워질 **Window**에 대한 설정을 적용시킬 **Form** 객체
6. **Show()**메서드 --> 새로운 **Window Form**를 띄워준다.

3-1. TTL(Text To Speech)



1. 참조추가에서 **system.Speech** 를 추가한다.





3-2. TTL(Text To Speech)

```
public partial class Form2 : Form
{
    SpeechSynthesizer voice;
    참조 1개
    public Form2()
    { InitializeComponent(); }
    참조 1개
    private void Form2_Load(object sender, EventArgs e)
    { voice = new SpeechSynthesizer(); }//load
    참조 1개
    private void btnSpeak_Click(object sender, EventArgs e)
    {
        try
        {
            voice.SetOutputToDefaultAudioDevice();
            voice.SelectVoice("Microsoft Heami Desktop");
            voice.SpeakAsync(textBox1.Text);
        }//
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }//catch
    }// speak
```

SpeechSynthesizer
:설치된 음성 합성 엔진의 기능에 액세스할 수 있습니다.
새 SpeechSynthesizer 개체를 만들 때 기본 시스템 음성을 사용합니다

SetOutputToDefaultAudioDevice()
: 기본 오디오 디바이스에 출력을 보내도록 SpeechSynthesizer 개체를 구성합니다.

SelectVoice(String)
: 특정 음성 이름으로 선택합니다.

SpeakAsync
메서드는 SpeakAsync 음성을 비동기적으로 생성합니다. 메서드는 개체의 SpeakAsync 내용이 말하기를 마칠 때까지 기다리지 않고 즉시 반환됩니다

Exception
: 예외 클래스의 기초 클래스로 사용할 수 있는 exception 클래스
: C#을 포함한 모든 .NET 프로그래밍 언어는
.NET Framework와 Exception 메카니즘에 따라 Exception을 처리
: 예외 처리는 이 Exception 객체를 기본으로 처리하게 됩니다.
: C#에서는 try, catch, finally라는 키워드를 사용하여 Exception을 핸들링

3-3. TTL(Text To Speech)

```
private void btnStop_Click(object sender, EventArgs e)
{
    try
    {
        voice.Pause();
    }//
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }//catch
}//stop

참조 1개
private void btnRestart_Click(object sender, EventArgs e)
{
    try
    {
      voice.Resume();
    }//
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }//catch
}//restart
```

Pause()
: SpeechSynthesizer 개체를 일시 정지합니다.

Resume()
: 일시 중지된 후 SpeechSynthesizer 개체를 다시 시작

MessageBox
: 흔히 프로그램을 사용할 때 발생하는 정보창/경고창 입니다.

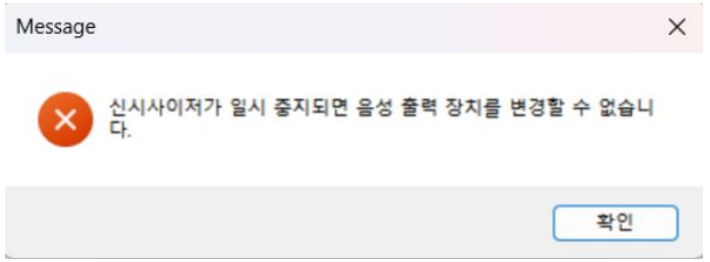
MessageBox.Show(arg1, arg2, arg3, arg4)

arg1 -> MessageBox Window 내용에 나타낼 텍스트를 의미합니다.

arg2 -> MessageBox Window 제목에 나타낼 텍스트를 의미합니다.

arg3 -> MessageBox 버튼 타입을 지정합니다.

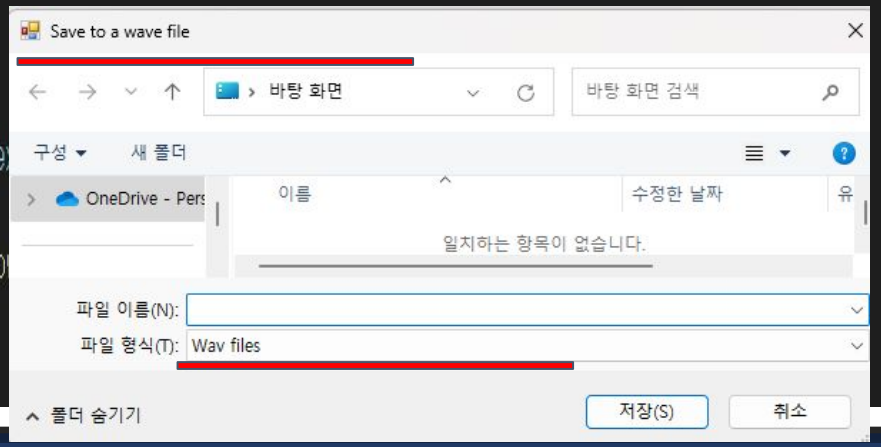
arg4 -> MessageBox Icon을 지정합니다.



3-4. TTL(Text To Speech)

```
using System.Windows.Forms;
using System.Speech.Synthesis;
using System.IO;

private void btnSave_Click(object
{
    try
    {
        using (SaveFileDialog sfd = new SaveFileDialog())
        {
            sfd.Filter = "Wav files|*.wav";
            sfd.Title = "Save to a wave file";
            if (sfd.ShowDialog() == DialogResult.OK)
            {
                FileStream fs = new FileStream(sfd.FileName, FileMode.Create, FileAccess.Write);
                voice.SetOutputToWaveStream(fs);
                voice.Speak(textBox1.Text);
            }//if
        }//using
    }//
    catch (Exception e
    {
        MessageBox.Sho
    }//catch
}//save
```



using(){ }
: using문을 메소드 안에 커넥트 형식으로 { } 안에서만 사용하고 나면
바로 dispose 되게끔 만들어 줌

SaveFileDialog 클래스
: 사용자에게 파일을 저장할 위치를 선택하라는 메시지를 표시
Filter
: 대화 상자에서 "파일 형식으로 저장" 또는 "파일 형식" 상자에 표시되는 선택 옵션을 결정하는 현재 파일 이름 필터 문자열을 가져오거나 설정
Title
: 파일 대화 상자 제목을 가져오거나 설정

FileStream(string FileName, FileMode, FileAccess)
: 지정된 경로, 생성 모드 및 읽기/쓰기 권한을 사용
using System.IO : 선언을 하고 나서 new 연산자를 이용하여 스트림 클래스의 파생 클래스들을 사용합니다.
DialogResult
: ShowDialog()를 호출하여 모달창을 띄우고
모달 창에서 [OK] 또는 [Cancel] 버튼을 클릭했는지를 알아냄

FileMode.Create
: 새로운 파일을 생성하며,만약 같은 파일이 있으면 덮어쓴다
FileAccess.Write
: 파일을 쓰기 전용(파일 내용 수정할 수 있는 권한)으로 사용한다.

voice.SetOutputToWaveStream()
: 오디오 스트림으로 출력하도록 신디사이저를 구성
Speak(String)
: 문자열의 내용을 동기적으로 말합니다

THANK YOU