

본 강의에서 수업자료로 이용되는 저작물은
저작권법 제25조 수업목적 저작물 이용 보상금제도에 의거,
한국복제전송저작권협회와 약정을 체결하고 적법하게 이용하고 있습니다.
약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로
수업자료의 재 복제, 대중 공개·공유 및 수업 목적 외의 사용을 금지합니다.

2024. 8. 30.

부천대학교·한국복제전송저작권협회

C#

9장 대화상자

4주차 학습 내용

* 1차시

- * 3주차 3차시 과제 피드백=> 동영상제공
- * 9장 대화상자
 - * 대화상자
 - * 모달 방식
 - * 모달리스 방식
 - * 메시지 상자
 - * 공통 대화상자

* 2차시

- * 9장 다중 폼
 - * 모달 방식

* 3차시

- * 4장 델리게이트, 이벤트
- * 9장 다중 폼
 - * 모달리스 방식

[클래스 멤버의 종류]

-자료 멤버

필드, 상수, 이벤트

-함수멤버

메서드, 생성자, 소멸자, 프러퍼티 (속성), 인덱서(색인자), 연산자 중복

-내포형 멤버

클래스형, 구조형, 열거형, 인터페이스형, 대리(델리게이트)형

[객체 지향 프로그램 구성]

-클래스 정의

-객체 선언

-객체 생성

-객체의 멤버 접근 및 활용

C# 프로그래밍



9. 대화상자



목차

- 대화상자
- 메시지 상자
- 공통 대화상자



대화상자

- 대화상자의 용도
 - 사용자와 애플리케이션간의 교량 역할 .
 - 주로 소량의 데이터를 입출력하기 위한 수단
- 대화상자의 생성방법
 - 모달(modal) 대화상자
 - 모달리스(modeless) 대화상자
- 대화상자의 종류
 - 메시지 대화상자
 - 사용자에게 간단한 메시지 표현.
 - 공통 대화상자
 - 윈도우 운영체제에서 기본적으로 제공.
 - 열기, 저장, 글꼴, 색, 인쇄, 페이지 설정 등.

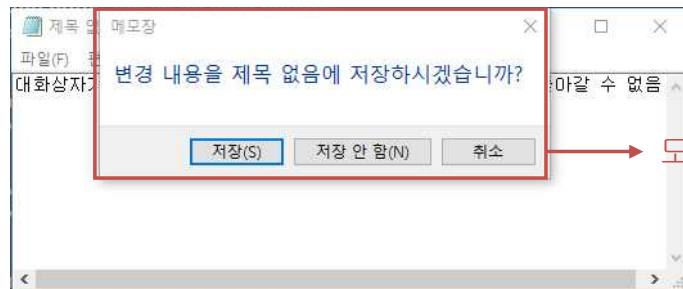


대화상자 – 모달 대화상자 [1/4]

- 대화상자가 종료되기 전에 대화상자를 띄운 애플리케이션으로 돌아갈 수 없음.
- 모달 대화상자 만드는 방법
 - Form 클래스의 멤버인 ShowDialog() 메소드 이용.
 - 모달 대화상자 만들기 예

```
Form2 form2 = new Form2();  
form2.ShowDialog(); // form2를 모달 방식으로 띄운다.
```

- 모달 대화상자 예
 - 메모장에서 편집내용을 저장하지 않고 종료할 때.



모달 대화상자



대화상자 – 모달 대화상자 [2/4]

- Form1에서 버튼을 클릭하여 Form2를 모달방식으로 띄우는 예제.

예제 9.1 [ex9_1_xxxx] : 모달 방식

★ Form1

1) 디자인



컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ModalApp
Button1 : button1	Text	Modal

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

2) 코드

```
private void button1_Click(object sender, EventArgs e) {  
    Form2 form2 = new Form2();  
    form2.ShowDialog();           // form2를 모달 방식으로 띄운다.  
}
```

[객체 지향 프로그램 구성]

- 클래스 정의
- 객체 선언
- 객체 생성
- 객체의 멤버 접근 및 활용

*Form2 추가 방법(Form2 클래스 정의)

메뉴 => 프로젝트 => 양식(Windows Form 추가)

*Form2 의 객체 선언과 생성

```
Form2 form2 = new Form2();
```

* form2 화면에 보이기

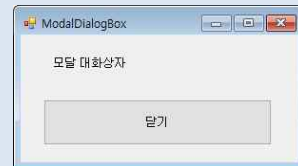
```
form2.ShowDialog();
```




대화상자 – 모달 대화상자 [3/4]

예제 9.1 [ex9_1_xxxx] : 모달 방식

★ Form2
1) 디자인



컨트롤 : (Name)	프로퍼티	값
Form : Form2	Text	ModalDialogBox
Label : label1	Text	모달 대화상자
Button1 : button1	Text	닫기

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

2) 코드

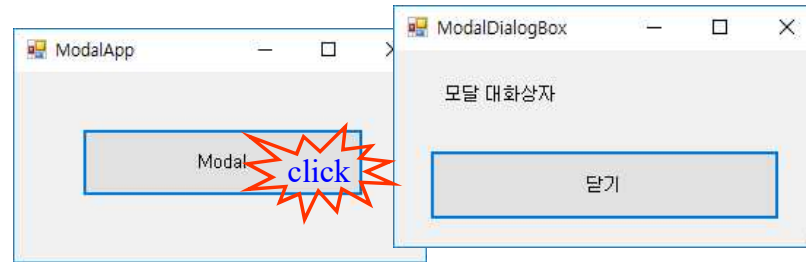
```
private void button1_Click(object sender, EventArgs e) {  
    close();  
}
```



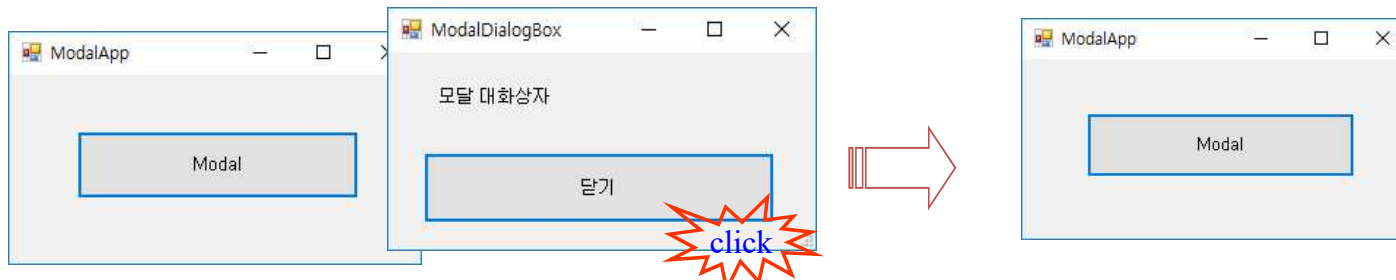
대화상자 - 모달 대화상자 [4/4]



① [Modal] 버튼 클릭



② 모달이므로 ModalApp 폼을 클릭하여도 돌아 갈 수 없음.



③ ModalDialogBox를 닫아야만 ModalApp 폼으로 돌아 갈 수 있음.

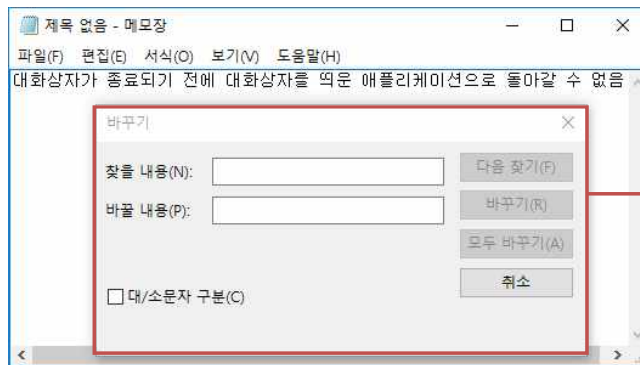


대화상자 – 모달리스 대화상자 [1/4]

- 현재 대화상자의 요구에 반응하지 않아도 다른 대화상자로 전환 가능.
 - 문자열 찾기, 검색, 도움말 기능 구현에 유용.
- 모달리스 대화상자 만드는 방법
 - Form 클래스의 멤버인 Show() 메소드 이용.
 - 모달리스 대화상자 만들기 예

```
Form2 form2 = new Form2();  
form2.Show(); // form2를 모달리스 방식으로 띄운다.
```

- 모달리스 대화상자 예
 - 메모장에서 문자열 바꾸기 대화상자.



모달리스 대화상자



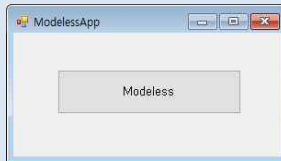
대화상자 – 모달리스 대화상자 [2/4]

- Form1에서 버튼을 클릭하여 Form2를 모달리스방식으로 띄우는 예제.

예제 9.2 [ex9_2_xxxx] :모달리스 방식

★ Form1

1) 디자인



컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ModelessApp
Button1 : button1	Text	Modeless

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

2) 코드

```
private void button1_Click(object sender, EventArgs e) {  
    Form2 form2 = new Form2();  
    form2.Show();           // form2를 모달리스 방식으로 띄운다.  
}
```

[객체 지향 프로그램 구성]

- 클래스 정의
- 객체 선언
- 객체 생성
- 객체의 멤버 접근 및 활용

*Form2 추가 방법(Form2 클래스 정의)

메뉴 => 프로젝트 => 양식(Windows Form 추가)

*Form2 의 객체 선언과 생성

```
Form2 form2 = new Form2();
```

* form2 화면에 보이기

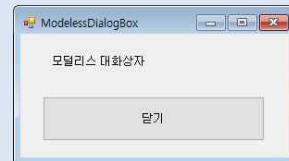
```
form2.Show();
```



대화상자 – 모달리스 대화상자 [3/4]

예제 9.2 [ex9_2_xxxx] :모달리스 방식

★ Form2
1) 디자인



컨트롤 : (Name)	프로퍼티	값
Form : Form2	Text	ModelessDialogBox
Label : label1	Text	모달리스 대화상자
Button1 : button1	Text	닫기
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

2) 코드

```
private void button1_Click(object sender, EventArgs e) {  
    Close();  
}
```

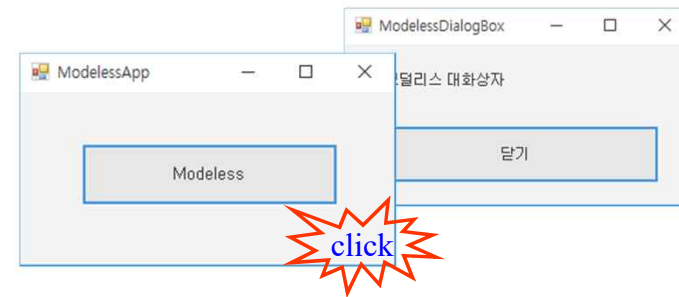


대화상자 - 모달리스 대화상자 [4/4]

실행 결과 :



① [Modeless] 버튼 클릭



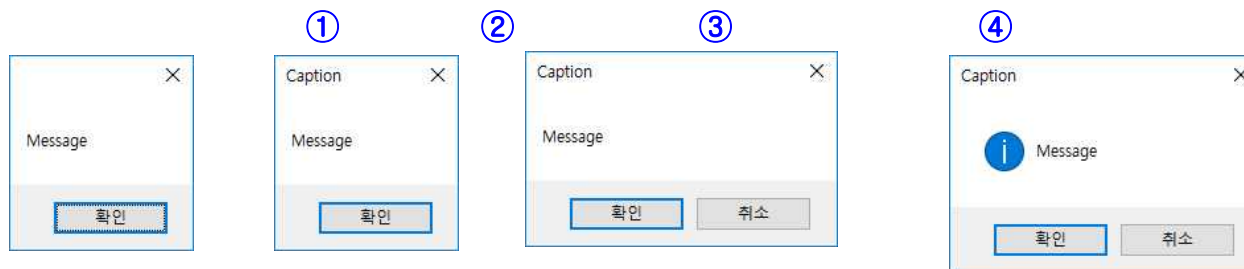
② 모달리스이므로 ModelessApp 폼으로
돌아 갈 수 있음.



메시지 상자

- 사용자에게 간단한 메시지를 전달할 때 사용
- MessageBox 클래스의 멤버인 Show() 메소드 이용
- 12개의 중복된 Show() 메소드 중 기본 형식

```
MessageBox.Show(message); ----- ①  
MessageBox.Show(message, caption); ----- ②  
MessageBox.Show(message, caption, buttonKind); ----- ③  
MessageBox.Show(message, caption, buttonKind, iconKind); ----- ④
```





메시지 상자 - 버튼

■ 버튼의 종류

■ MessageBoxButtons 열거형의 멤버로서 5가지




기호상수 (멤버이름)	순서 값	버튼 모양	설 명
OK	0		OK 버튼
OKCancel	1	 	OK, Cancel 버튼
AbortRetryIgnore	2	  	Abort, Retry, Ignore 버튼
YesNoCancel	3	  	Yes, No, Cancel 버튼
YesNo	4	 	Yes, No 버튼
RetryCancel	5	 	Retry, Cancel 버튼



메시지 상자 – 아이콘

■ 아이콘의 종류

- MessageBoxIcon 열거형의 멤버로서 9개의 기호상수.
- 아이콘의 모양은 4가지

기호상수 (멤버이름)	순서 값	아이콘 모양	설 명
None	0		기호 없음.
Error	16		빨간색 배경의 원 안에 흰색 X가 포함된 기호.
Hand			
Stop			
Question	32		풍선 안에 물음표가 포함된 기호.
Exclamation	48		노란색 배경의 삼각형 안에 느낌표가 있는 기호.
Warning			
Asterisk	64		풍선 안에 소문자 i가 포함된 기호.
Information			



메시지 상자 – 기본 버튼 설정

■ 기본 버튼

- 메시지 상자가 활성화 될 때 초기에 입력포커스를 갖는 버튼.
- 기본 버튼을 명시하지 않으면 첫 번째 버튼이 기본 버튼.

■ 기본 버튼 설정 방법

- MessageBoxDefaultButton 열거형 멤버를 매개변수로 갖는 Show() 메소드 이용.

```
Show(message, caption, buttonKind, iconKind, MessageBoxDefaultButton);
```

- MessageBoxDefaultButton 열거형

기호상수(멤버이름)	순서 값	설 명
Button1	0x000	왼쪽을 기준으로 첫 번째 버튼을 기본으로 설정
Button2	0x100	왼쪽을 기준으로 두 번째 버튼을 기본으로 설정
Button3	0x200	왼쪽을 기준으로 세 번째 버튼을 기본으로 설정



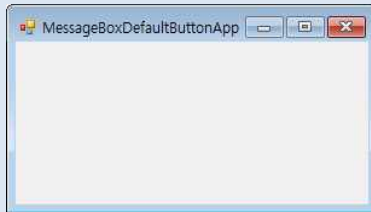
메시지 상자 예제 [1/2]

■ 폼을 클릭하여 메시지 박스를 띄우는 예제.

예제 9.6 [ex9_6_xxxx] : 메시지 상자

```
private void Form1_Click(object sender, EventArgs e) {
```

1) 디자인



컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	MessageBoxDefaultButtonApp
컨트롤 : (Name)	이벤트	메소드명
Form : Form1	Click	Form1_Click()

```
    DialogResult ans = MessageBox.Show("버튼 선택", "제목", MessageBoxButtons.YesNo);  
  
    if (ans == DialogResult.Yes)  
        label1.Text = "yes";  
    else  
        label1.Text = "no";  
}
```

2) 코드

```
private void Form1_Click(object sender, EventArgs e) {  
    MessageBox.Show("MessageBoxDefaultButton", "Title Bar",  
        MessageBoxButtons.YesNoCancel,  
        MessageBoxIcon.Question,  
        MessageBoxDefaultButton.Button2);  
}
```



메시지 상자 예제 [2/2]

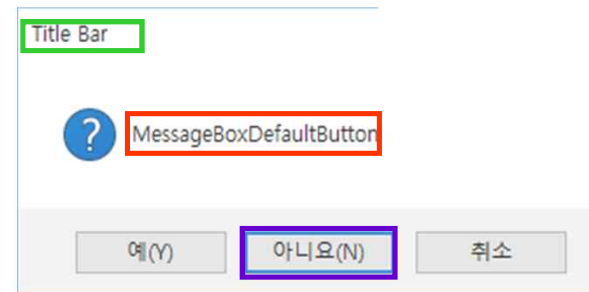
실행 결과 :



① MessageBoxDefaultButtonApp 폼 클릭

☺ 참고

```
MessageBox.Show("MessageBoxDefaultButton",  
                "Title Bar",  
                MessageBoxButtons.YesNoCancel,  
                MessageBoxIcon.Question,  
                MessageBoxDefaultButton.Button2);
```





공통 대화상자

- 윈도우 운영체제에서 기본적으로 제공.
- 정형화된 대화상자
- CommonDialog 클래스의 파생 클래스
 - FileDialog
 - OpenFileDialog (파일 열기)
 - SaveFileDialog (파일 저장)
 - FontDialog (글꼴)
 - ColorDialog (색)
 - PrintDialog (인쇄)
 - PageSetupDialog (페이지 설정)
 - FolderBrowserDialog (폴더 탐색/생성)



공통 대화상자 – 열기 대화상자

- 드라이브, 폴더, 파일 확장자를 설정하여 원하는 형식의 파일을 찾을 수 있는 기능 제공.
- OpenFileDialog 컴포넌트의 주요 프로퍼티

프로퍼티	설명
FileName	대화상자에서 선택된 절대경로 형태로 구성된 파일명.
FileNames	Multiselect 프로퍼티가 참으로 설정된 경우에 파일명들을 나타내는 스트링 배열.
Filter	콤보 상자에 표시될 문자열(파일 형식)과 해당 파일 형식을 선택할 때 사용하게 될 확장자. “파일형식1 확장자1 파일형식2 확장자2...” 형식으로 명시.
FilterIndex	대화상자에서 현재 선택된 Filter 프로퍼티의 인덱스.
InitialDirectory	대화상자에 표시하는 초기 디렉토리.
RestoreDirectory	종료 전에 초기 디렉토리로 되돌아갈 지의 여부.
Multiselect	대화상자에서 여러 파일들을 선택할 수 있는 지의 여부.

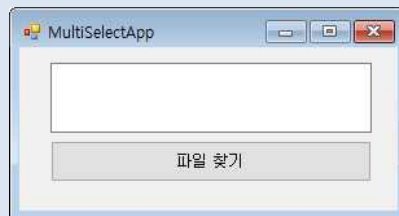


공통 대화상자 – 열기 대화상자

- 버튼을 클릭하여 열기대화상자를 띄우고 선택한 파일의 경로와 이름을 텍스트 상자에 출력하는 예제.

예제 9.7 [ex9_7_xxxx] : 열기 대화상자 1

1) 디자인



컴포넌트 : (Name)	프로퍼티	값
OpenFileDialog : openFileDialog1	ShowColor	

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	OpenFileDialog
TextBox : textbox1	Text	

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()



공통 대화상자 – 열기 대화상자

2) 코드

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.InitialDirectory = @"C:\W";
    openFileDialog1.Filter = "텍스트 파일(*.txt)|*.txt|모든 파일(*.*)|*.*";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.RestoreDirectory = true;
    openFileDialog1.ShowDialog();
    textBox1.Text = openFileDialog1.FileName
}
```

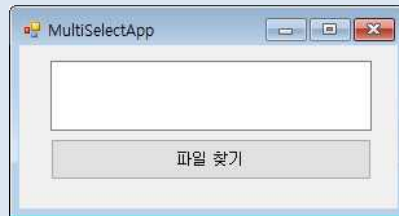



공통 대화상자 – 열기 대화상자

- 버튼을 클릭하여 열기대화상자를 띄우고 선택한 파일의 경로와 이름을 텍스트 상자에 출력하는 예제.

예제 9.8 [ex9_8_xxxx] : 열기 대화상자 2

1) 디자인



컴포넌트 : (Name)	프로퍼티	값
OpenFileDialog : openFileDialog1	ShowColor	True

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	MultiSelectApp
Label : label1	Text	파일 찾기
TextBox : textbox1	Text	
	Multiline	True

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()



공통 대화상자 – 열기 대화상자

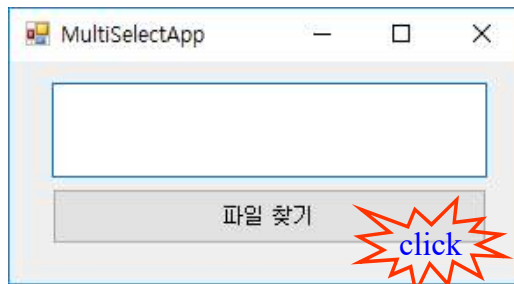
2) 코드

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.InitialDirectory = @"C:\W";
    openFileDialog1.Filter = "텍스트 파일(*.txt)|*.txt|모든 파일(*.*)|*.*";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.RestoreDirectory = true;
    openFileDialog1.Multiselect = true;
    openFileDialog1.ShowDialog();
    foreach(string strTmp in openFileDialog1.FileNames)
    {
        textBox1.Text += strTmp;
        textBox1.Text += "\r\n";
    }
}
```

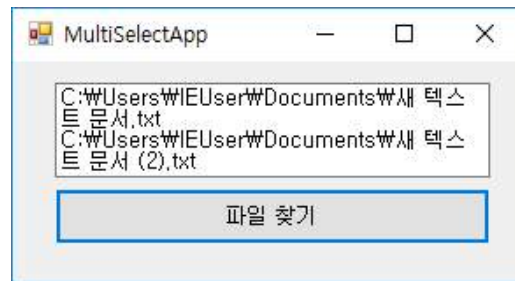


공통 대화상자 - 열기 대화상자

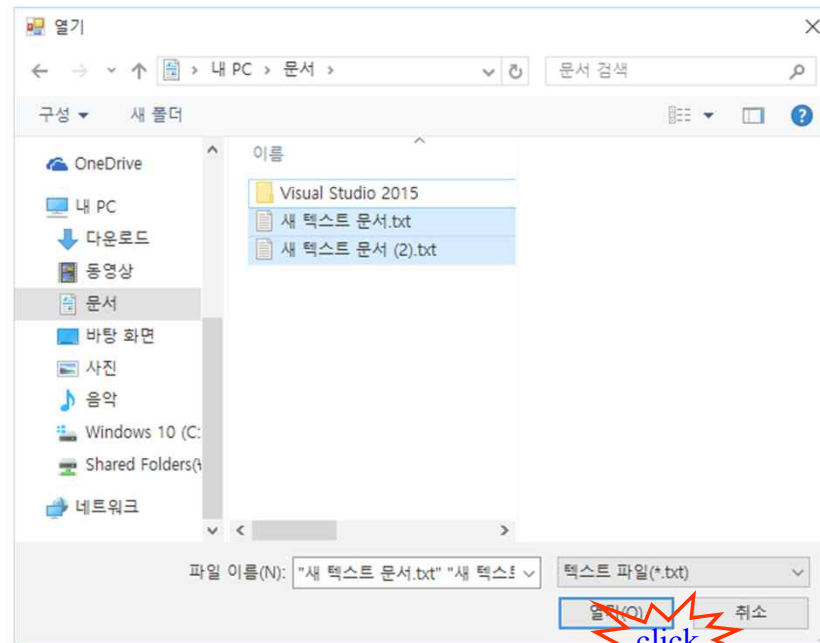
실행 결과 :



① [파일 찾기] 버튼 클릭.



③ 텍스트 박스에 선택된 파일의 리스트가 출력됨.



② 2개 이상의 파일을 선택하여 [열기]버튼 클릭.



공통 대화상자 – 글꼴 대화상자 [1/3]

- 글꼴, 글자의 크기, 글자의 색상, 형태 등을 설정할 수 있는 기능 제공.
- FontDialog 컴포넌트의 주요 프로퍼티

프로퍼티	설 명
Color	글꼴 대화상자에서 선택된 색상
Font	글꼴 대화상자에서 선택한 글꼴 및 글꼴의 크기
ShowApply	글꼴 대화상자에 [적용] 버튼의 추가 여부 제어 True : 추가, False : 추가하지 않음
ShowColor	글꼴 대화상자에서 색 콤보박스 추가 여부 제어 True : 추가, False : 추가하지 않음

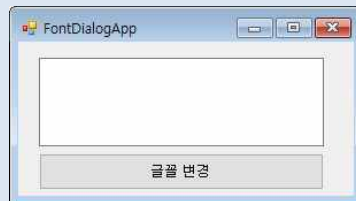


공통 대화상자 – 글꼴 대화상자 [2/3]

- 버튼을 클릭하여 글꼴대화상자를 띄우고 텍스트 상자의 글 속성들을 변경시키는 예제.

예제 9.9 [ex9_9_xxxx] : 글꼴 대화상자

1) 디자인



컴포넌트 : (Name)	프로퍼티	값
FontDialog : fontDialog1	ShowColor	True

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	FontDialogApp
Label : label1	Text	글꼴 변경
TextBox : textbox1	Text	
	Multiline	True

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

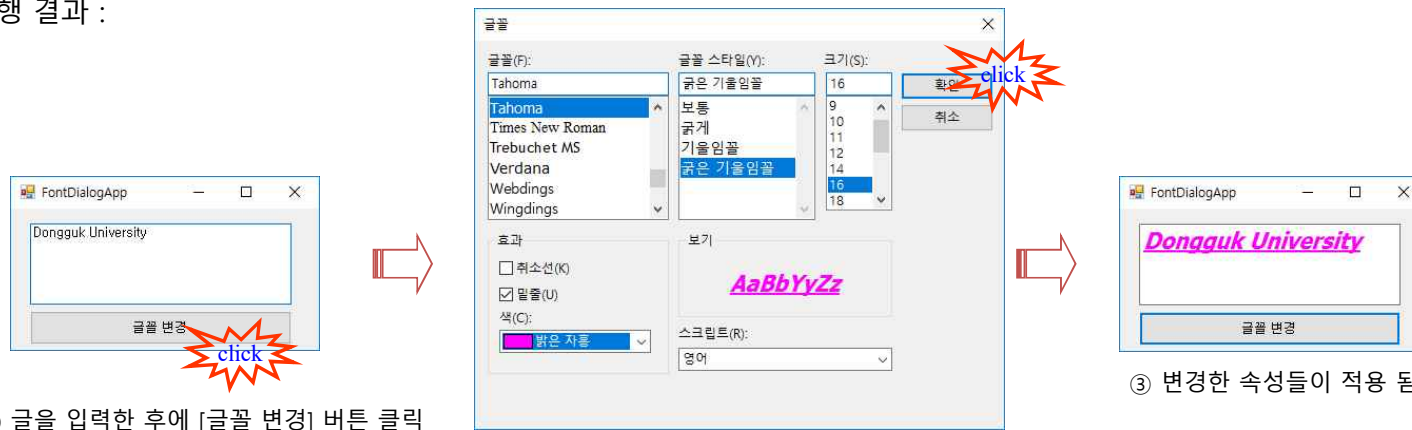


공통 대화상자 – 글꼴 대화상자 [3/3]

2) 코드

```
private void button1_Click(object sender, EventArgs e) {  
    fontDialog1.ShowDialog();  
    textBox1.Font = fontDialog1.Font;  
    textBox1.ForeColor = fontDialog1.Color;  
}
```

실행 결과 :





공통 대화상자 – 색 대화상자 [1/4]

- 색상표에서 기본 색을 선택하거나 사용자 지정 색을 만들어 사용할 수 있는 기능 제공.
- ColorDialog 컴포넌트 이용





공통 대화상자 – 색 대화상자 [2/4]

- button1을 클릭하여 폼의 배경색을 변경하고 button2를 클릭하여 버튼의 배경색을 변경하는 예제.

예제 9.10 [ex9_10_xxxx] : 색 대화상자

1) 디자인



컴포넌트 : (Name)	프로퍼티	값
ColorDialog : colorDialog1		
컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ColorDialogApp
Button : button1	Text	폼 색상 변경
Button : button2	Text	버튼 색상 변경

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()
Button : button2	Click	button2_Click()



공통 대화상자 – 색 대화상자 [3/4]

2) 코드

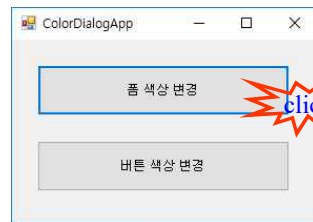
```
private void button1_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    this.BackColor = colorDialog1.Color;    // 폼의 배경 색
}

private void button2_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    button1.BackColor = colorDialog1.Color; // 버튼의 배경 색
    button2.BackColor = colorDialog1.Color;
}
```



공통 대화상자 – 색 대화상자 [4/4]

실행 결과 :



③ [버튼 색상 변경] 버튼을 클릭하여 위와 같이 색 선택.

② 색을 선택하고 [확인] 버튼 클릭.



공통 대화상자 – 인쇄 대화상자 [1/4]

- 인쇄할 프린터, 인쇄 범위, 인쇄 매수 등을 선택할 수 있는 기능 제공.
- 인쇄 대화상자 만들기
 - PrintDialog 컴포넌트와 두 개의 클래스가 더 필요.
 - System.Drawing.Printing.PrinterSettings (기본프린터 설정)
 - System.Drawing.Printing.PrintDocument (출력물 설정)
 - PrintPage 이벤트에 PrintPageEventHandler 델리게이트 등록.
 - System.Drawing.Printing 네임스페이스 추가.



공통 대화상자 – 인쇄 대화상자 [2/4]

- 버튼을 클릭하여 인쇄 대화상자를 띄우고 테스트 상자의 내용을 프린터로 출력하는 예제.

예제 9.11 [ex9_11_xxxx] : 인쇄 대화상자

1) 디자인



컴포넌트 : (Name)	프로퍼티	값
PrintDialog : printDialog1		
컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	PrintDialogApp
TextBox : textBox1	Multiline	True
	Text	
Button : button1	Text	출력
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()



공통 대화상자 – 인쇄 대화상자 [3/4]

2) 코드

```
private void button1_Click(object sender, EventArgs e){
    PrinterSettings printer = new PrinterSettings();
    PrintDocument pd = new PrintDocument();
    printDialog1.PrinterSettings = printer;
    printDialog1.Document = pd;
    // PrintPage 이벤트는 Print() 메소드가 호출되기 직전 발생
    pd.PrintPage += new PrintPageEventHandler(this.pd_PrintPage);
    DialogResult result = printDialog1.ShowDialog();
    if (result==DialogResult.OK){
        pd.Print();
    }
}

private void pd_PrintPage(object sender, PrintPageEventArgs e){
    string text = textBox1.Text;
    Font printFont = new Font("Arial", 10, FontStyle.Regular);
    e.Graphics.DrawString(text, printFont, Brushes.Black, 10, 10);
}
```

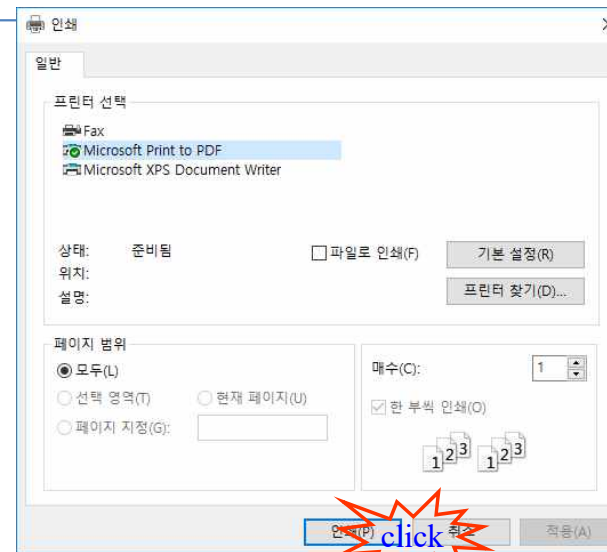


공통 대화상자 – 인쇄 대화상자 [4/4]

실행 결과 :



- ① 텍스트 상자에 글자를 입력하고 [출력] 버튼 클릭.



- ② 프린터를 선택하고 [확인] 버튼 클릭하면 문서가 출력됨.

C# 프로그래밍



9. 다중 폼(정보 전달)



대화상자

- 대화상자의 용도
 - 사용자와 애플리케이션간의 교량 역할 .
 - 주로 소량의 데이터를 입출력하기 위한 수단.
- 대화상자의 생성방법
 - 모달(modal) 대화상자
 - 모달리스(modeless) 대화상자
- 대화상자의 종류
 - 메시지 대화상자
 - 사용자에게 간단한 메시지 표현.
 - 공통 대화상자
 - 윈도우 운영체제에서 기본적으로 제공.
 - 열기, 저장, 글꼴, 색, 인쇄, 페이지 설정 등.



다중 폼에서 정보 전달

■ 대화상자의 용도

- **사용자와 애플리케이션간의 교량 역할** .
- 주로 **소량의 데이터**를 입출력하기 위한 수단.

■ 다중 폼에서 정보 전달 방식

■ **모달(modal)** 대화상자

- 다중폼1 [multiform1_xxxx] : 다중 폼 모달 방식 1(Form1 주체)
- 다중폼2 [multiform2_xxxx] : 다중 폼 모달 방식 2(Form2 주체)

■ **모덜리스(modeless)** 대화상자

- 다중폼3 [multiform3_xxxx] : 다중 폼 모덜리스 방식 3
 - => 이벤트를 이용
 - => 델리게이트



다중 폼 : 정보 교환 1 : 모달 방식(Form1 주체)

- Form1에서 버튼을 클릭하여 Form2에 Form1의 label의 정보 출력

다중폼1 [multiform1_xxxx] : 다중 폼 모달 방식 1

★ Form1

1) 실행



★ Form2



2) Form1 소스

public partial class Form1 : Form

```
{
    private void button1_Click(object sender, EventArgs e)
    {
        Form2 dlg = new Form2();
        dlg.LabelX = label1.Left;
        dlg.LabelY = label1.Top;
        dlg.LabelText = label1.Text;
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            label1.Left = dlg.LabelX;
            label1.Top = dlg.LabelY;
            label1.Text = dlg.LabelText;
        }
        dlg.Dispose();
    }
}
```

ShowDialog 메서드 기능

- 1) 대화 상자 띄우기
- 2) 대화상자의 실행 결과를 DialogResult 타입의 열거형으로 리턴하여 대화 상자에서 호출원으로 어떤 결과를 보고 할 때 리턴값이 사용된다.



다중 폼 : 정보 교환 1 : 모달 방식(Form1 주체)

- Form1(부모폼)이 정보의 전달 및 리턴 주체로 Form2(대화상자)의 속성으로 정보 전달과 결과를 리턴 받는다.

다중폼1 [multiform1_xxxx] : 다중 폼 모달 방식 1

★ Form2 속성 설정

AcceptButton : button1, CancelButton : button2

★ Form2의 button1, button2 속성 설정 : DialogResult :OK, Cancel

3) Form2 소스

```
public partial class Form2 : Form
{
    public int LabelX
    {
        get { return Convert.ToInt32(textBox1.Text); }
        set { textBox1.Text = value.ToString(); }
    }
    public int LabelY
    {
        get { return Convert.ToInt32(textBox2.Text); }
        set { textBox2.Text = value.ToString(); }
    }
    public string LabelText
    {
        get { return textBox3.Text; }
        set { textBox3.Text = value; }
    }
}
```

- 공개된 필드를 쓸 수도 있지만 안전성이 떨어지므로 프로퍼티를 사용하는 것이 좋다.




다중 폼 : 정보 교환 2 : 모달 방식(Form2 주체)

- Form1에서 버튼을 클릭하여 Form2에 Form1의 label의 정보 출력
- Form1(부모폼)는 대화상자만 호출하고 자신이 오너라는 것만 Form2(대화상자)에게 알려 준다.

다중폼2 [multiform2_xxxx] : 다중 폼 모달 방식 2

★ Form1


1) 실행



2) Form1 소스

```
public partial class Form1 : Form
{
    private void button1_Click(object sender, EventArgs e)
    {
        Form2 dlg = new Form2();
        dlg.Owner = this;
        dlg.ShowDialog();
        dlg.Dispose();
    }
}
```

★ Form2



- `public System.Windows.Forms.Label label1;`
=> 캡슐화의 원칙에 위배하며 위험하다.



다중 폼 : 정보 교환 2 : 모달 방식(Form2 주체)

- Form2(대화상자)가 정보의 전달 및 리턴 주체

다중폼2 [multiform2_xxxx] : 다중 폼 모달 방식 2

```
3) Form2 소스
public partial class Form2 : Form
{
    private void Form2_Load(object sender, EventArgs e)
    {
        Form1 Parent = (Form1)Owner;
        textBox1.Text = Parent.label1.Left.ToString();
        textBox2.Text = Parent.label1.Top.ToString();
        textBox3.Text = Parent.label1.Text;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Form1 Parent = (Form1)Owner;
        Parent.label1.Left=Convert.ToInt32(textBox1.Text);
        Parent.label1.Top = Convert.ToInt32(textBox2.Text);
        Parent.label1.Text = textBox3.Text;
    }
}
```

- 대화 상자는 프로퍼티만 제공하고 부모가 정보를 입출력하는 것이 좋다.

C# 프로그래밍



9. 다중 폼(정보 전달) 모델리스 방식



다중 폼에서 정보 전달

■ 대화상자의 용도

- **사용자와 애플리케이션간의 교량 역할** .
- 주로 **소량의 데이터**를 입출력하기 위한 수단.

■ 다중 폼에서 정보 전달 방식

■ **모달(modal)** 대화상자

- 다중폼1 [multiform1_xxxx] : 다중 폼 모달 방식 1(Form1 주체)
- 다중폼2 [multiform2_xxxx] : 다중 폼 모달 방식 2(Form2 주체)

■ **모덜리스(modeless)** 대화상자

- 다중폼3 [multiform3_xxxx] : 다중 폼 모덜리스 방식 3
 - => 이벤트를 이용
 - => 델리게이트



다중 폼 : 정보 교환 3 : 모달리스 방식

- Form2 대화상자를 모달리스 대화 상자로 사용 시 항상 열어둠

다중폼3 [multiform3_xxxx] : 다중 폼 모달리스 방식 3

★ Form1

1) 실행



★ Form2



2) Form1 소스

```
public partial class Form1 : Form
{
    private Form2 dlg;
    private void OnApply(object sender, EventArgs e)
    {
        label1.Left = dlg.LabelX;
        label1.Top = dlg.LabelY;
        label1.Text = dlg.LabelText;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (dlg == null || dlg.IsDisposed)
        {
            dlg = new Form2();
            dlg.Owner = this;
            dlg.Apply += new EventHandler(OnApply);
            dlg.LabelX = label1.Left;
            dlg.LabelY = label1.Top;
            dlg.LabelText = label1.Text;
            dlg.Show();
        }
    }
}
```




다중 폼 : 정보 교환 3 : 모델리스 방식

■ Form2 대화상자를 모델리스 대화 상자로 사용시 항상 열어둠

다중폼3 [multiform3_xxxx] : 다중 폼 모델리스 방식 3

★ Form2의 button1, button2 속성 설정 : DialogResult :None

3) Form2 소스

```
public partial class Form2 : Form
{
    public int LabelX
    {
        get { return Convert.ToInt32(textBox1.Text); }
        set { textBox1.Text = value.ToString(); }
    }
    public int LabelY
    {
        get { return Convert.ToInt32(textBox2.Text); }
        set { textBox2.Text = value.ToString(); }
    }
    public string LabelText
    {
        get { return textBox3.Text; }
        set { textBox3.Text = value; }
    }
    public event EventHandler Apply;
    private void button1_Click(object sender, EventArgs e)
    {
        if (Apply != null)
        {
            Apply(this, new EventArgs()); //매개변수 타입이 같아야 함!
        }
    }
    private void button2_Click(object sender, EventArgs e)
    {
        Dispose();
    }
}
```



델리게이트

■ 델리게이트

- 메서드를 참조하기 위한 기법
- 하나의 클래스에서 다른 것은 필요 없고, 단지 특정 객체의 메서드 만 사용 하고자 할 때 사용하는 기법
- 이벤트와 스레드를 처리하는데 주로 사용
- C/C++ 언어에서 함수 포인터와 유사한 기능을 갖고 있지만 포인터보다는 객체 지향적이며 타입 안정적임
- 묵시적으로 System.Delegate 클래스를 상속한 클래스형으로 간주

```
[modifiers] delegate returnType DelegateName(parameterList);
```

■ 델리게이트 예제

```
delegate void SampleDelegate(int param); // 델리게이트 정의  
class DelegateClass {  
    public void DelegateMethod(int param) { // 델리게이트할 메소드  
        // ...  
    }  
}
```



이벤트

- 이벤트(event)
 - 사용자 행동에 의해 발생하는 사건
 - 어떤 사건이 발생한 것을 알리기 위해 보내는 메시지
 - C#에서는 이벤트 개념을 프로그래밍 언어 수준에서 지원
- 이벤트 처리기(event handler)
 - 발생한 이벤트를 처리하기 위한 메소드
- 이벤트-주도 프로그래밍(event-driven programming)
 - 이벤트와 이벤트 처리기를 통하여 객체에 발생한 사건을 다른 객체에 통지하고 그에 대한 행위를 처리하도록 시키는 구조를 가짐
 - 각 이벤트에 따른 작업을 독립적으로 기술
 - 프로그램의 구조가 체계적/구조적이며 복잡도를 줄일 수 있음
- 정의 형태

```
[event-modifier] event DelegateType EventName;
```