

제6장

델리게이트(Delegate)

변영철

1. 델리게이트 의미

- 델리게이트(delegate)
- 대리인
- 대신 사용하는 것
- 무엇 대신? 함수 대신

delegate

- ▶ C# 응용에서 이벤트를 처리하는 기법으로 Delegate가 있음.
- ▶ 많은 코드가 미리 준비되어 있고(응용 프레임워크) 우리가 원하는 부분만 고쳐서 사용할 수 있음.

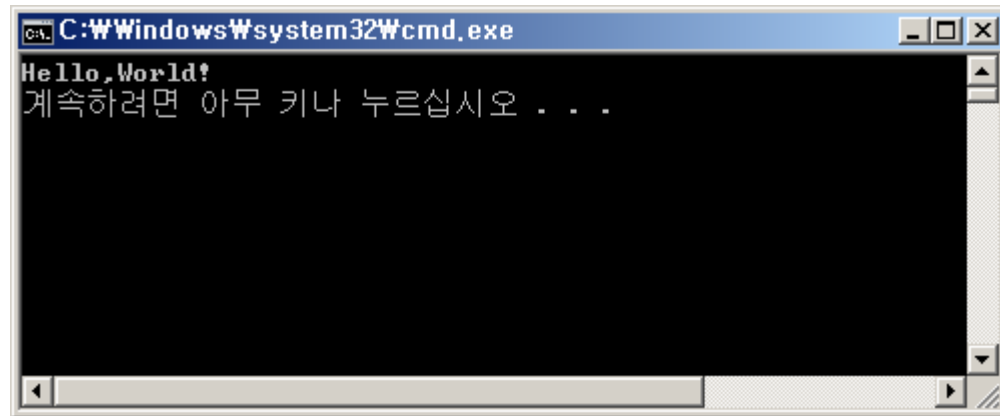
2. 프로젝트 생성

- 새로운 빈 프로젝트 Delegate 생성
- 프로젝트 | 새 항목 추가
 - Delegate.cs 파일

3. 코드 편집 및 작성

- 기본 코드를 “Hello,World!” 표시하도록 변경

```
public class Delegate
{
    public static void Main() {
        System.Console.WriteLine("Hello,World!");
    }
}
```



3. 코드 편집 및 작성

- “클릭!” 표시하도록 변경

```
public class Delegate
{
    public static void Main() {
        System.Console.WriteLine("클릭!");
    }
}
```



4. 코드 추상화와 함수

- xxx 함수로 코드 추상화

```
using System;
```

```
public class Delegate  
{
```

```
    public void xxx() {  
        Console.WriteLine("클릭!");  
    }
```

```
    public static void Main() {  
        xxx(); //실행될까??  
    }
```

```
}
```

4. 코드 추상화와 함수

- 클래스는 객체(gildong) 만들라고 있는 것
- gildong이에게 xxx 해달라고 하자.

```
using System;
```

```
public class Delegate
{
    public void xxx() {
        Console.WriteLine("클릭!");
    }

    public static void Main() {
        Delegate gildong = new Delegate();
        gildong.xxx();
    }
}
```

5. 데이터 추상화 클래스

- Base 클래스 작성

```
public class Base  
{  
  
}
```


5. 데이터 추상화 클래스

- xxx 함수를 Base로 옮기기

```
public class Base
{
    public void xxx() {
        Console.WriteLine("클릭!");
    }
}
```

5. 데이터 추상화 클래스

- 기존 코드 수정

```
public class Delegate
{
    public static void Main() {
        Base gildong = new Base();
        gildong.xxx();
    }
}
```

6. 델리게이트로 호출하기

- 델리게이트 이용하기(xxx 대신 Click 사용)

```
public class Delegate
{
    public static void Main() {
        Base gildong = new Base();
        gildong.Click(); //Click은 xxx 함수의 델리게이트
    }
}
```

- ▶ xxx 멤버 함수를 호출했었는데 이제는 xxx 대신 Click 사용(호출)
- ▶ 만일 이렇게 하더라도 오류가 없다면
- ▶ Click은 xxx 멤버 함수의 델리게이트(대신 사용하는 것)

6. 델리게이트로 호출하기

- C언어 함수 포인터 변수 : 델리게이트와 유사

```
void xxx() {  
    printf("Hello,World!");  
}
```

```
void main()  
{  
    void xxx()* Click = null; //void (*click) (); 이 맞는 문법  
  
    Click= xxx;  
    Click();  
}
```

6. 델리게이트로 호출하기

- 함수 포인터 변수 Click, 객체
- Click은 변수 혹은 객체
- 객체를 만들려면 클래스가 있어야
- 클래스 이름이 DelegateClass일 경우

```
using System;
```

```
public class Base
{
    public DelegateClass Click = null;
    public void xxx() {
        Console.WriteLine("클릭!");
    }
}
```

6. 델리게이트로 호출하기

- 자료형 선언: delegate 키워드를 사용

```
using System;
```

```
public class Base  
{
```

```
    public delegate void DelegateClass();  
    public DelegateClass Click = null;
```

```
    public void xxx() {  
        Console.WriteLine("클릭!");  
    }
```

```
}
```

리턴값이 없고(void)
파라미터가 없는() 함
수를 대신 실행할 수
있음을 의미.

6. 델리게이트로 호출하기

- 델리게이트 객체생성 및 xxx 함수 설정

```
public class Base
{
    public delegate void DelegateClass();
    public DelegateClass Click = null;

    public Base() {
        Click = new DelegateClass(xxx);
    }

    public void xxx() {
        Console.WriteLine("클릭!");
    }
}
```

6. 델리게이트로 호출하기

- 컴파일 및 실행



6. 델리게이트로 호출하기

- 생성자 함수에서
델리게이트 Click을 xxx와
연결하지 않을 경우에는
어떤 일이?



```
public class Base
{
    public delegate void DelegateClass();
    public DelegateClass Click = null;

    public void xxx() {
        System.Console.WriteLine("Hello,World!");
    }
}

public class Delegate
{
    public static void Main() {
        Base gildong = new Base();
        gildong.Click();
    }
}
```

6. 델리게이트로 호출하기

- 이런 경우를 핸들링 하려면?

```
public class Base
{
    public delegate void DelegateClass();
    public DelegateClass Click = null;

    public void xxx() {
        System.Console.WriteLine("Hello,World!");
    }
}
```

```
public void OnClick() {
    if (Click != null)
        Click();
}
```

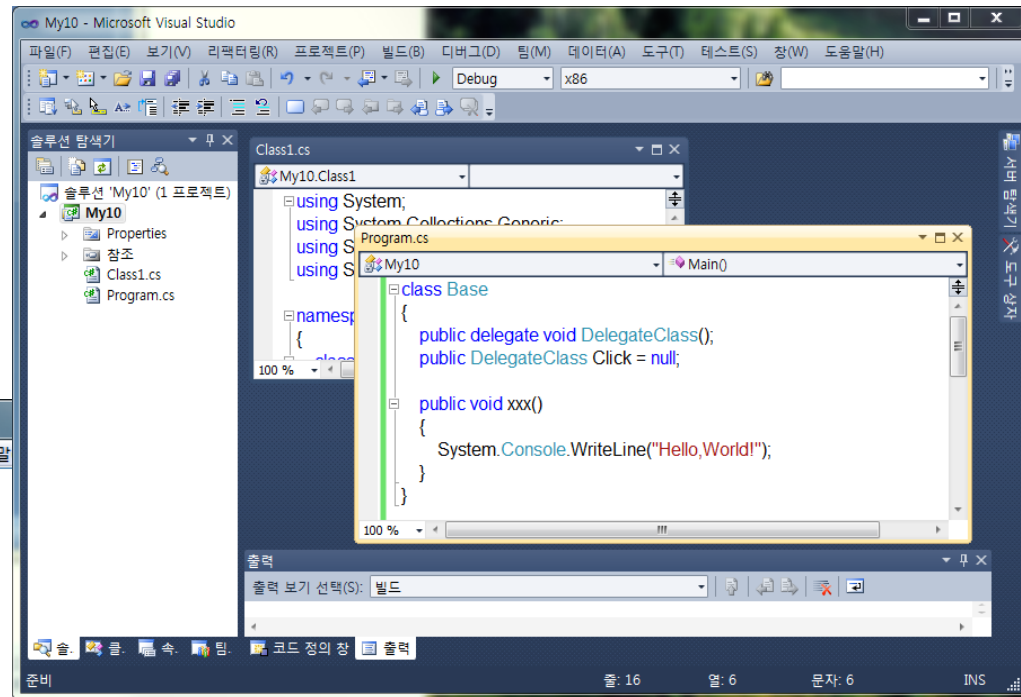
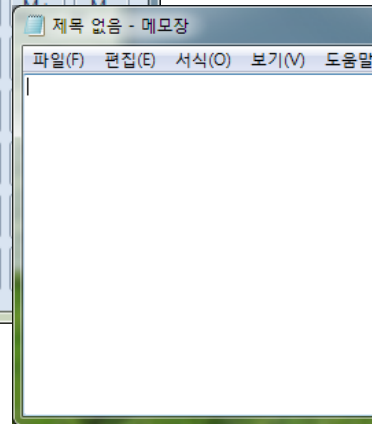
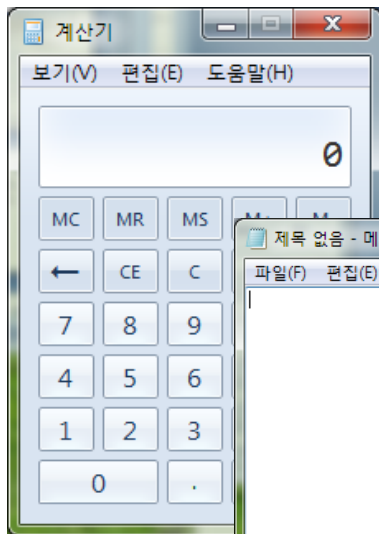
```
public class Delegate
{
    public static void Main() {
        Base gildong = new Base();
        gildong.OnClick();
    }
}
```

7. 델리게이트와 이벤트

| | |
|---------------|-------------------------------------|
| Click | 델리게이트, 델리게이트 객체, 델리게이트 인스턴스, 이벤트 |
| DelegateClass | 델리게이트 형(type) |
| xxx | 핸들러 함수 |
| OnClick | 이벤트 Click을 fire 하는 함수 |

7. 델리게이트와 이벤트

- 윈도우 운영체제 특징
 - GUI : 응용 프로그램이 서로 비슷
 - 멀티태스킹 기능
 - 이벤트 처리 기능

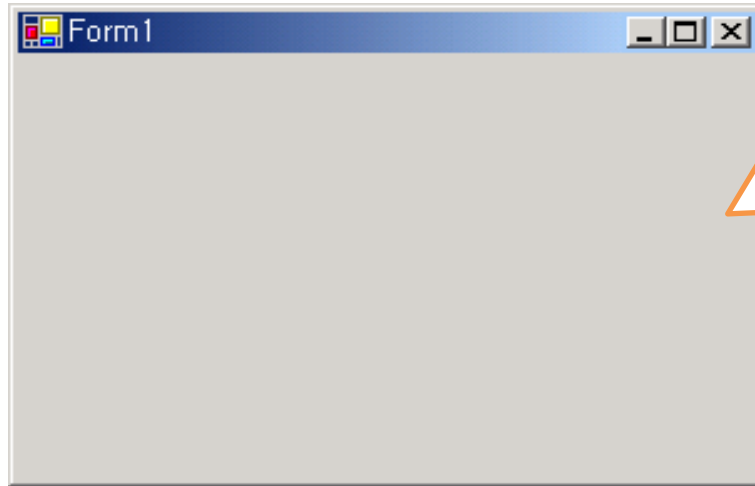


따라서 수많은 중복되는 코드들이 존재한다.

→ 라이브러리로 만들어 놓고 필요할 때 사용하면 좋겠다.

7. 델리게이트와 이벤트

- 폼 윈도우와 이벤트



1. 마우스로 폼 윈도우 클릭
2. .NET 런타임(runtime)은 폼 윈도우 객체의 OnClick이라는 가상 멤버 함수 호출
3. OnClick 메소드는 Click 이벤트 발생 (fire)

- OnClick 도움말

- 도움말 > 도움말 보기 메뉴 선택

Visual Studio 2010 - Windows Internet Explorer

http://msdn.microsoft.com/query/dev10,qu Google

즐거찾기

Visual Studio 2010

홈 라이브러리 학습 다운로드 지원 커뮤니티 로그인 | 한국(한국어) 기본 설정

Form

MSDN Library
↑ 개발 도구 및 언어
Visual Studio 2010
Visual Studio
Visual Studio Application Lifecycle Mana

커뮤니티 콘텐츠

이 항목을 보강하려면 코트 샘플 및
팁을 추가하십시오.


더 보기...

Visual Studio 2010

Visual Studio 2010

이 콘텐츠는 높은 품질 표준에 맞게 수작업으로 번역된 것입니다.이 페이지와
원본 영어 콘텐츠를 동시에 보려면 "기본 설정"을 클릭하고 클래식을 보기 기
본 설정으로 선택합니다.

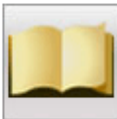
이 문서에서는 Visual Studio를 사용하여 데스크톱 응용 프로그램과 웹 응용 프로그
램을 만드는 방법을 설명하는 리소스의 링크를 제공합니다.



Visual Studio 2010 소개

Visual Studio 2010에 대해 알아보기

- Visual Studio 2010 시작 페이지
- Visual Studio 2010 제품의 중요 정보

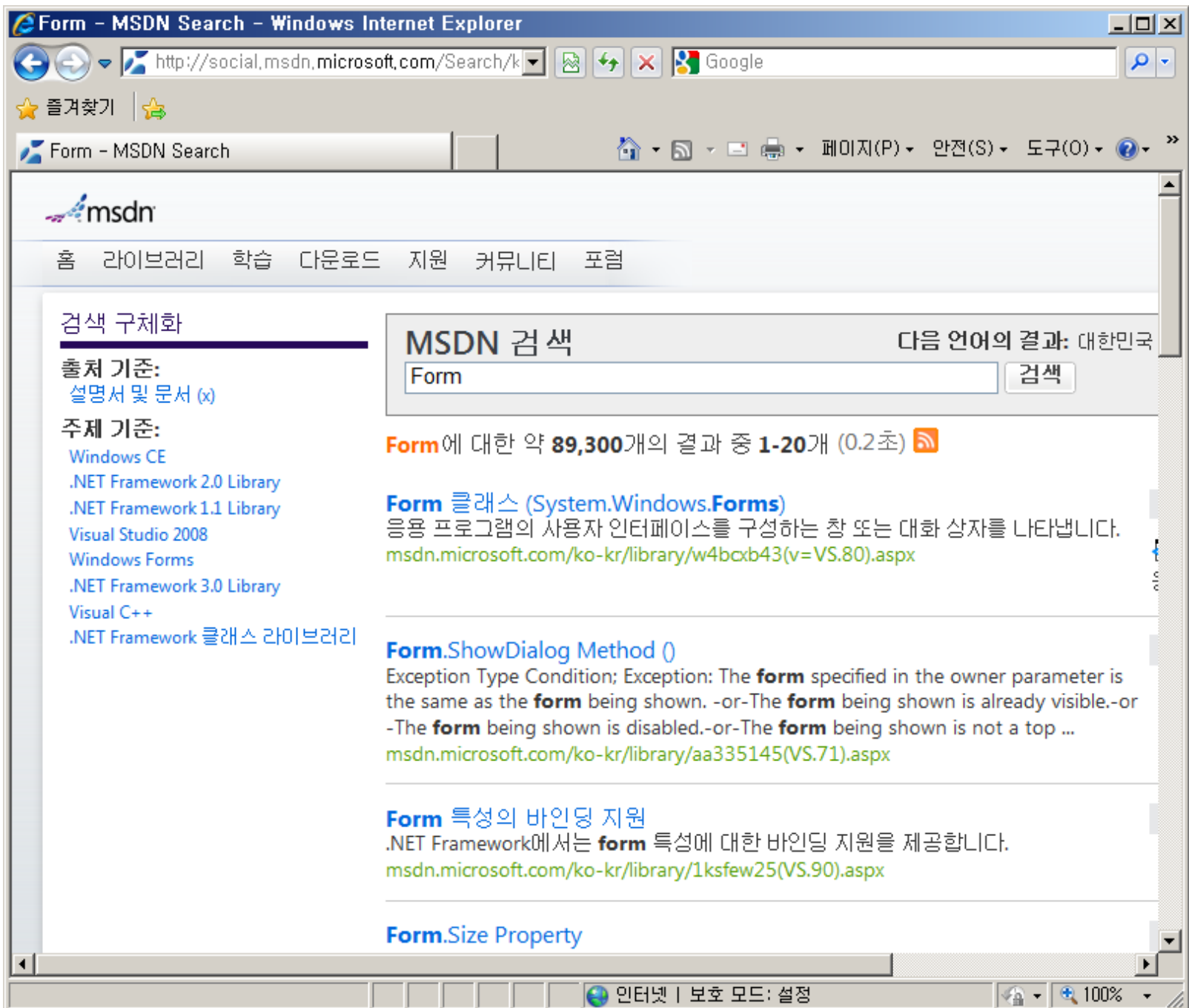


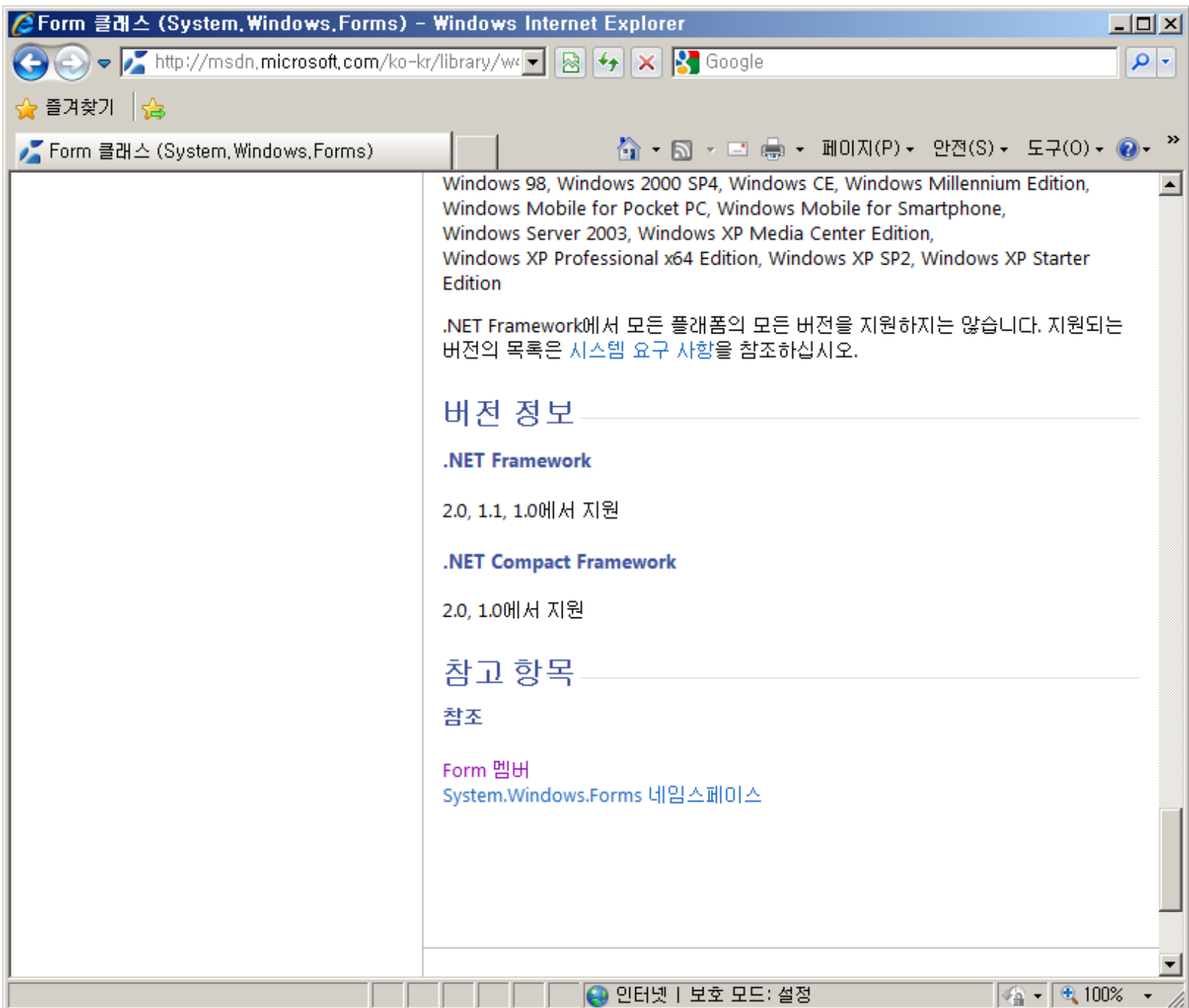
설명서

Visual Studio 2010 설명서에서 Visual Studio를 사용하여 응용 프로
그램을 만드는 방법을 배울 수 있습니다.

- Visual Studio

인터넷 | 보호 모드: 설정 100%





Form 멤버 (System.Windows.Forms) - Windows Internet Explorer

http://msdn.microsoft.com/ko-kr/library/sy Google

즐거찾기

Form 멤버 (System.Windows.Forms)

| | | |
|--|---------------------------|--|
| | | 속됨) |
| | OnChangeUICues | ChangeUICues 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnClick | Click 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnClientSizeChanged | ClientSizeChanged 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnClosed | Closed 이벤트를 발생시킵니다. |
| | OnClosing | Closing 이벤트를 발생시킵니다. |
| | OnContextMenuChanged | ContextMenuChanged 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnContextMenuStripChanged | ContextMenuStripChanged 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnControlAdded | ControlAdded 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnControlRemoved | ControlRemoved 이벤트를 발생시킵니다. (Control에서 상속됨) |
| | OnCreateControl | 재정의되었습니다. CreateControl 이벤트를 발생시킵니다. |
| | OnCursorChanged | CursorChanged 이벤트를 발생시킵니다. (Control에서 상속됨) |

인터넷 | 보호 모드: 설정 100%

Form 멤버 (System.Windows.Forms) - Windows Internet Explorer

http://msdn.microsoft.com/ko-kr/library/sy Google

즐거찾기

Form 멤버 (System.Windows.Forms)

Public 이벤트

| 이름 | 설명 |
|---|---|
|  Activated | 폼이 코드에서 활성화되거나 사용자에게 의해 활성화될 때 발생합니다. |
|  AutoSizeChanged | |
|  AutoValidateChanged | |
|  BackColorChanged | BackColor 속성 값이 변경되면 발생합니다.(Control에서 상속됨) |
|  BackgroundImageChanged | BackgroundImage 속성 값이 변경되면 발생합니다.(Control에서 상속됨) |
|  BackgroundImageLayoutChanged | BackgroundImageLayout 속성이 변경되면 발생합니다.(Control에서 상속됨) |
|  BindingContextChanged | BindingContext 속성 값이 변경되면 발생합니다.(Control에서 상속됨) |
|  CausesValidationChanged | CausesValidation 속성 값이 변경되면 발생합니다.(Control에서 상속됨) |
|  ChangeUICues | 포커스나 키보드 UI(사용자 인터페이스) 큐가 변경되면 발생합니다.(Control에서 상속됨) |
|  Click | 컨트롤을 클릭하면 발생합니다.(Control에서 상속됨) |
|  ClientSizeChanged | ClientSize 속성 값이 변경되면 발생합니다.(Control에서 상속됨) |

인터넷 | 보호 모드: 설정 100%

8. 델리게이트가 있는 이유

- public delegate **void** DelegateClass**0**;
- 리턴값이 없고(void) 파라미터가 없는 모든 함수를 대신해서 사용 가능함
- 핸들러 함수 이름이 아직 결정되지 않아서 델리게이트로 호출
- 개발자는 나중에 핸들러 함수(xxx)를 만들어 델리게이트에 연결
- 즉, **델리게이트가 있는 이유는** 응용 프레임워크에서 아직 존재하지 않는 핸들러 함수를 호출하도록 하기 위하여

9. Base 클래스 라이브러리화

- 앞으로 여러분이 어떤 프로그램을 작성하더라도 항상 Base라는 클래스를 작성해야 한다면
 - 매번 작성하는 것 보다는 차라리 **미리 만들어 놓고** 라이브러리로 사용하는 것이 바람직함.
 - 따라서 Base 클래스를 라이브러리로 제공하는 클래스라고 **가정**

9. Base 클래스 라이브러리화

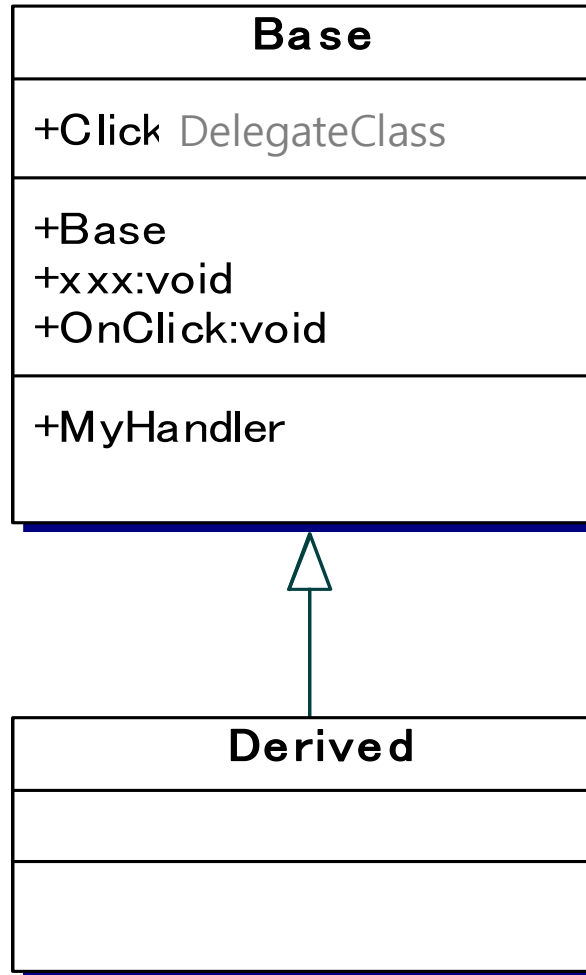
- 문제점
 - xxx라는 핸들러 함수 이름은 프로그래머가 결정하는 것이므로 Base 클래스 안에 기술되어 있으면 Base는 라이브러리가 될 수 없음.
- xxx 관련 코드를 파생(derived) 클래스로

```
class Derived : Base
{
    public void xxx() {
        Console.WriteLine("클릭!");
    }
}
```

10. 비하인드 코드/클래스 Derived

- Derived 클래스 작성
 - Derived : Base
 - Base 클래스 안에 작성된 코드가 고스란히 Derived 클래스 안으로 들어옴(재사용, 상속)
 - 따라서 Base 대신 Derived 클래스로 객체 만들어도 동일함
 - 개념적으로 볼 때 Derived 클래스 뒤쪽(behind)에 Base 클래스가 있는 모양이며, 따라서 Base를 **비하인드(behind) 클래스**, 혹은 비하인드 코드라고 부름. Derived는 프론트(front) 클래스

10. 비하인드 코드/클래스 Derived



11. Base 클래스 라이브러리화

```
class Derived : Base
{
    public Derived() {
        Click += new DelegateClass(xxx);
    }

    public void xxx() {
        Console.WriteLine("클릭!");
    }
}
```

라이브러리화 할 수 없는 코드들을 Derived로 밀어버림.

11. Base 클래스 라이브러리화

```
class My
{
    static void Main(string[] args) {
        Derived gildong = new Derived();
        gildong.OnClick();
    }
}
```

12. 가상 함수를 이용한 이벤트 처리

- Base 클래스의 OnClick을 가상 함수로 만들기

```
public virtual void OnClick() {  
    if(Click != null)  
        Click();  
}
```

가상(virtual) 함수, 마음에 들지 않으면 재정의하라. 그러면 새로 정의한 함수가 실행되고 원래 함수는 없는 것과 같이 상상속의, 가상속의 가상(virtual) 함수가 될 것이다.

12. 가상 함수를 이용한 이벤트 처리

- Derived 클래스에서 오버라이딩

```
public class Derived : Base
{
    public Derived() {
        Click += new DelegateClass(xxx);
    }

    public void xxx() {
        Console.WriteLine("클릭!");
    }

    public override void OnClick() {
    }
}
```

12. 가상 함수를 이용한 이벤트 처리

```
class My
{
    static void Main(string[] args) {
        Base gildong = new Derived();
        gildong.OnClick();
    }
}
```

12. 가상 함수를 이용한 이벤트 처리

- 기존에 있는 가상함수 OnClick이 전혀 마음에 들지 않으면...

```
public override void OnClick() {  
    Console.WriteLine("-----");  
    Console.WriteLine("클릭!");  
    Console.WriteLine("-----");  
}
```

12. 가상 함수를 이용한 이벤트 처리

- 기존에 있는 가상함수 OnClick가 불충분하여 보완하고자 한다면, 다음과 같이 기존의 OnClick 함수도 호출함.

```
public override void OnClick()
{
    Console.WriteLine("-----");
    base.OnClick();
    Console.WriteLine("-----");
}
```

12. 가상 함수를 이용한 이벤트 처리

- 따라서 이벤트가 발생할 경우 무엇인가를 하고 싶으면
- 다음과 같이 두 가지 방법 중 하나를 이용
 - 핸들러 함수(xxx)를 작성
 - 가상 함수(OnClick)를 오버라이딩

13. 코드 다듬기

- 클래스 이름 바꾸기
 - Base → Form
 - DelegateClass → EventHandler

14. Application 응용 클래스 작성

- Main 함수 내의 코드를 Run으로 추상화

```
public class My
{
    public void Run()
    {
        Form gildong = new Derived();
        gildong.OnClick();
    }

    public static void Main()
    {
        Run();
    }
}
```

14. Application 응용 클래스 작성

- 객체 정의 및 호출

```
public class My
{
    public void Run()
    {
        Form gildong = new Derived();
        gildong.OnClick();
    }

    public static void Main()
    {
        My cheolsu = new My();
        cheolsu.Run();
    }
}
```

14. Application 응용 클래스 작성

- 정적 멤버로 선언

```
public class My
{
    public static void Run() {
        Form gildong = new Derived();
        gildong.OnClick();
    }

    public static void Main() {
        Run();
    }
}
```

14. Application 응용 클래스 작성

- 독립된 클래스로 작성 → 라이브러리

```
public class Application
{
    public static void Run() {
        Form gildong = new Derived();
        gildong.OnClick();
    }
}
```

```
public class My
{
    public static void Main() {
        Application.Run();
    }
}
```

15. Application 클래스 라이브러리화

- 문제점

- 우선, Derived 클래스는 프로그래머가 임의로 정한 xxx를 포함하고 있으므로 라이브러리화 할 수 없음
- 그러한 Derived를 Application 클래스에서 사용하고 있으므로 Application 클래스 또한 현재로는 라이브러리화할 수 없음

15. Application 클래스 라이브러리화

- 라이브러리가 될 수 있을까?

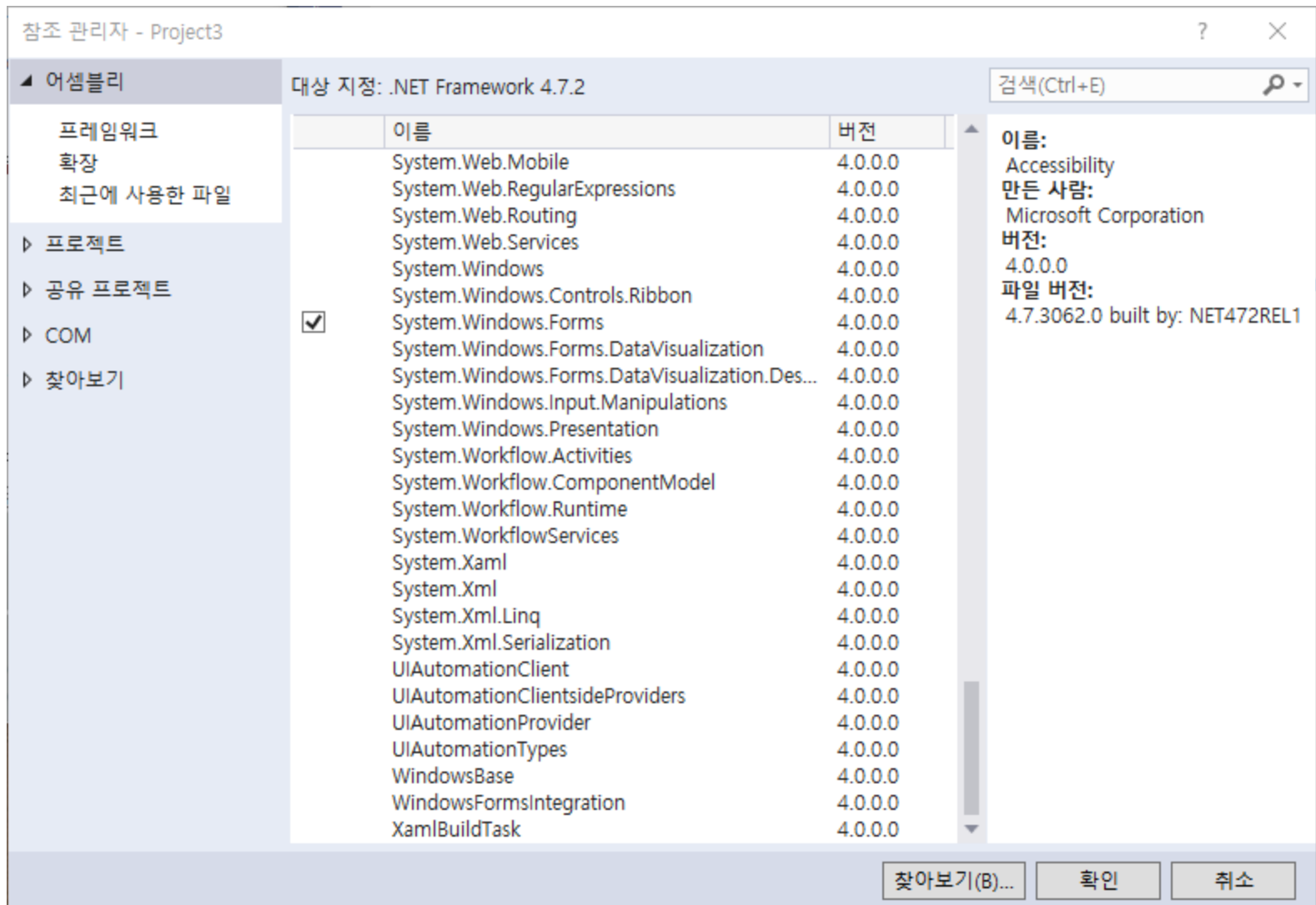
```
public class Application {  
    public static void Run(Form gildong)  
    {  
        gildong.OnClick();  
    }  
}
```

```
public class My {  
    public static void Main()  
    {  
        Application.Run(new Derived());  
    }  
}
```

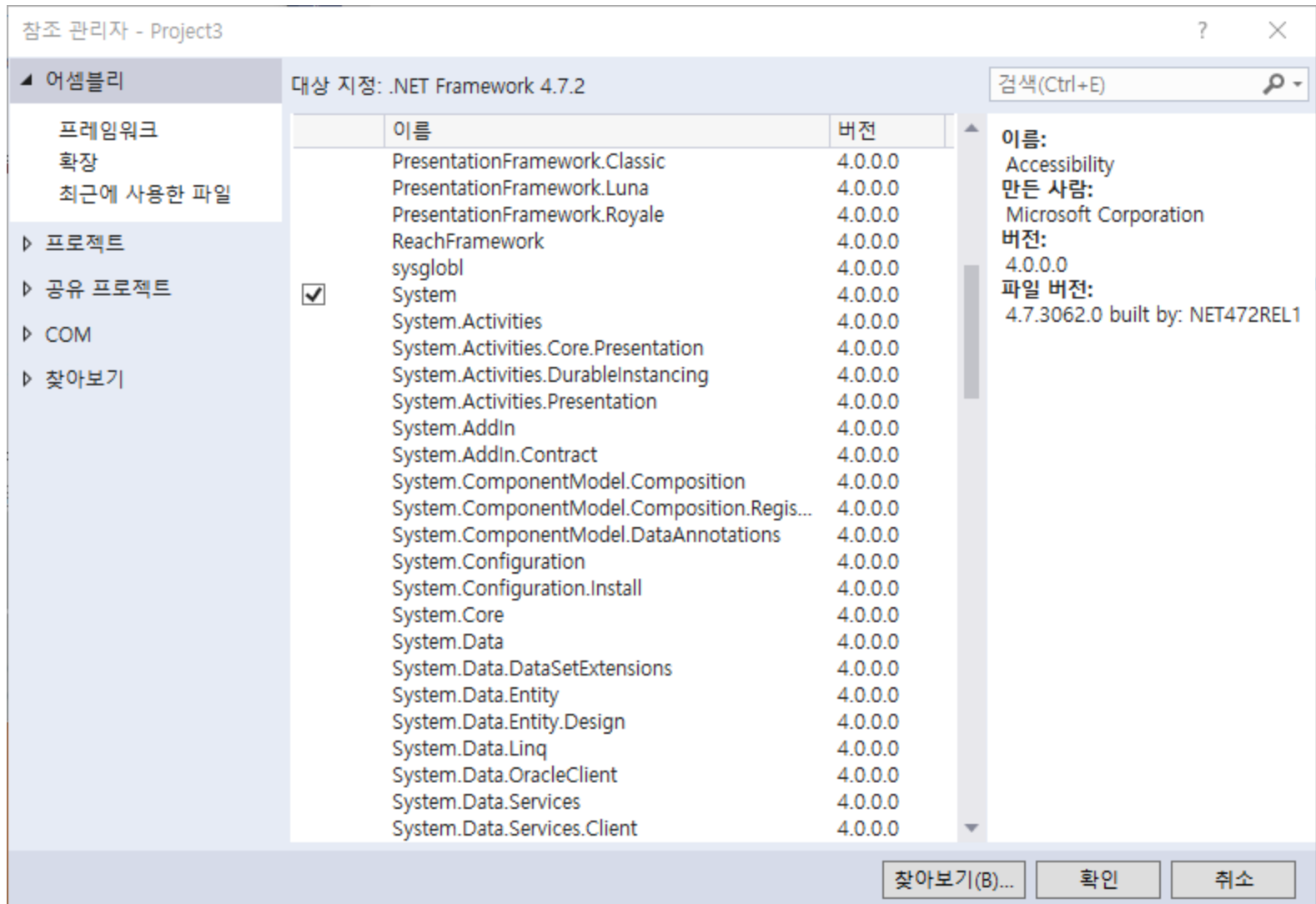
16. C# 클래스 라이브러리 이용하기

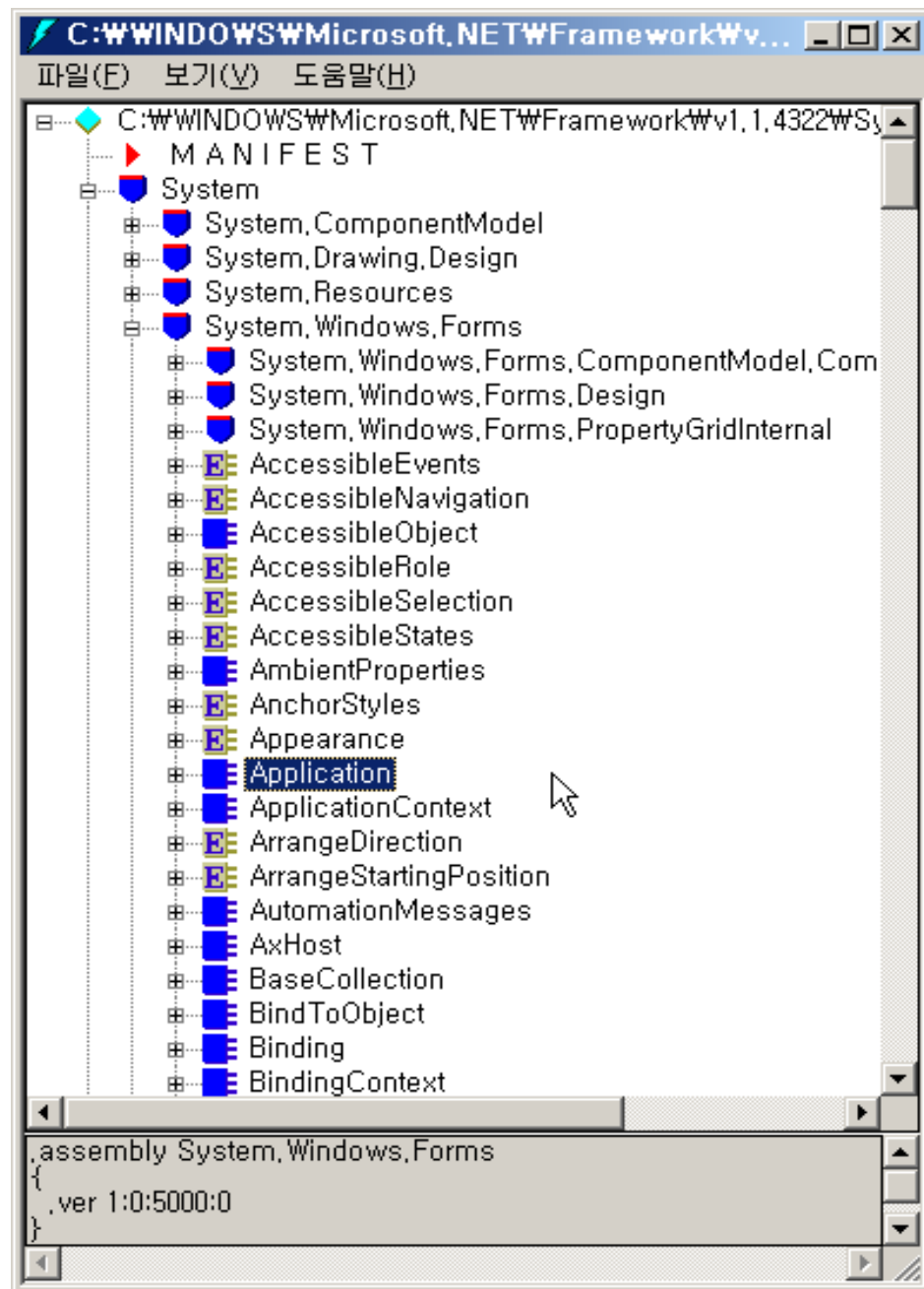
- Form, Application 클래스는 이미 닷넷 라이브러리에 존재함
- 따라서 힘들게 작성하기 보다는 이를 이용할 수 있음.
- 프로젝트 | 참조추가 메뉴 선택
 - Windows.System.Forms
 - System

15. C# 클래스 라이브러리 이용하기



15. C# 클래스 라이브러리 이용하기





16. 라이브러리에 있는 클래스 사용하기

- Form, Application 클래스가 이미 있으니 우리가 만든 클래스는 제거하자.
- 아래 코드 추가하기
`using System.Windows.Forms;`
- EventHandler 선언 고치기
`public delegate void EventHandler(Object o, EventArgs e);`

코드를 위와 같이 고쳤다. 그 의미는?

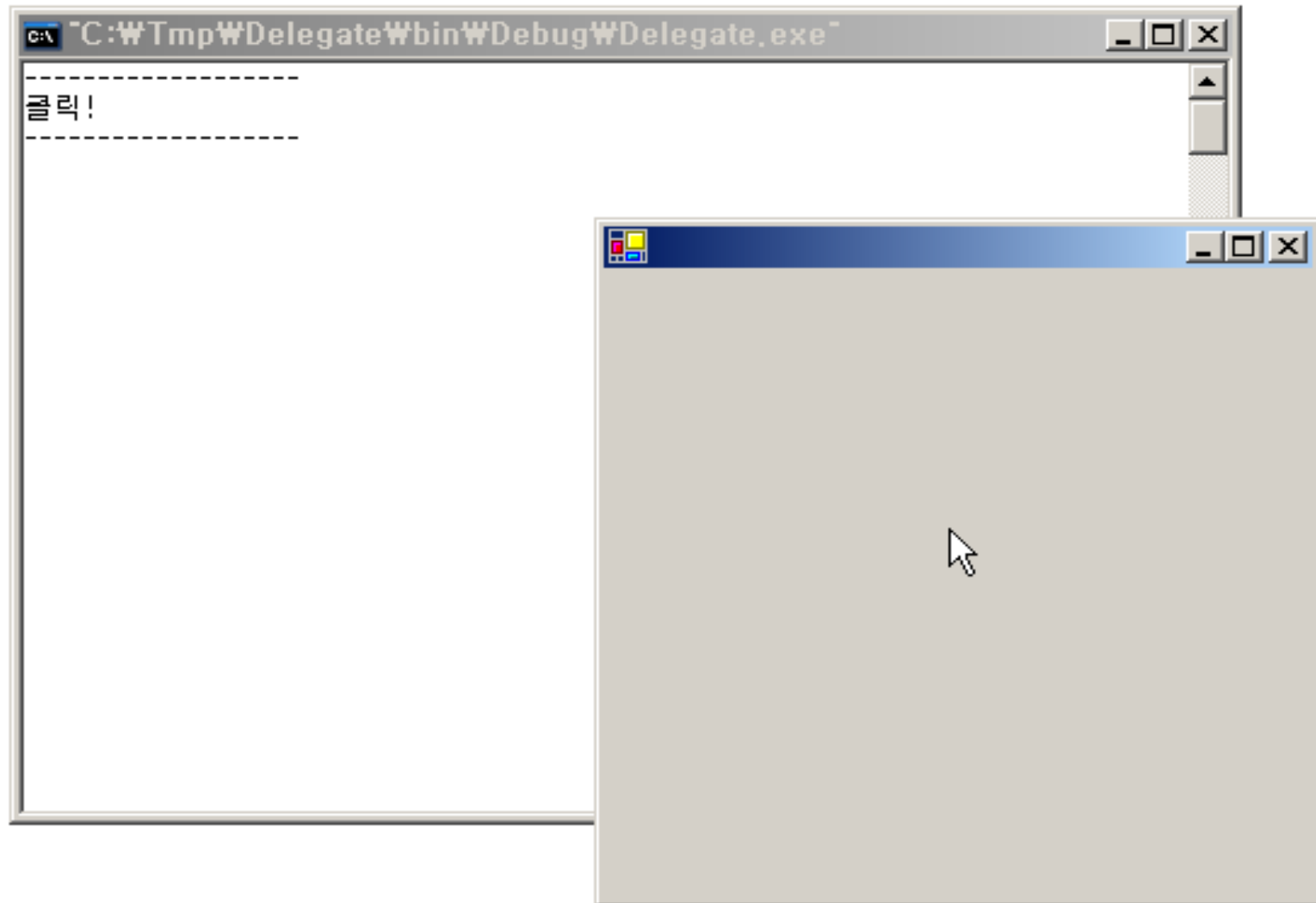
16. C# 클래스 라이브러리 이용하기

- 핸들러 함수 xxx 수정

```
public void xxx(Object o, EventArgs e)
{
    Console.WriteLine("클릭!");
}
```

- OnClick 가상함수 수정

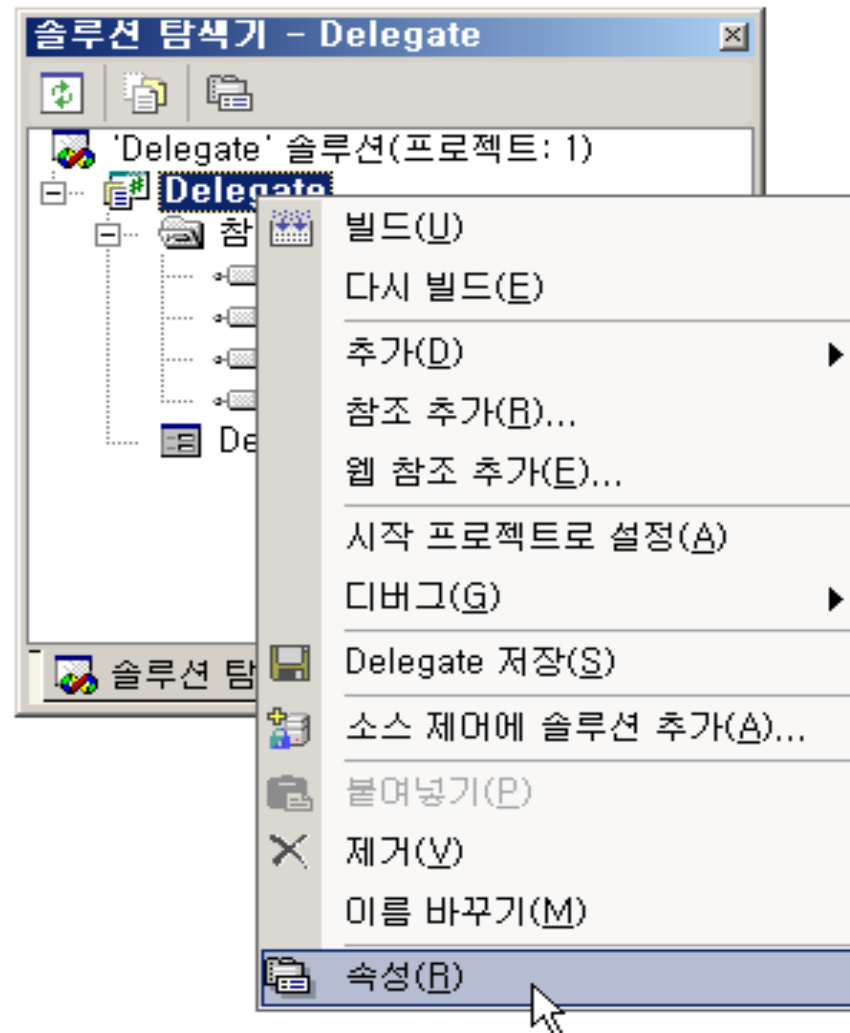
```
protected override void OnClick(EventArgs e)
{
    Console.WriteLine("-----");
    base.OnClick(e);
    Console.WriteLine("-----");
}
```



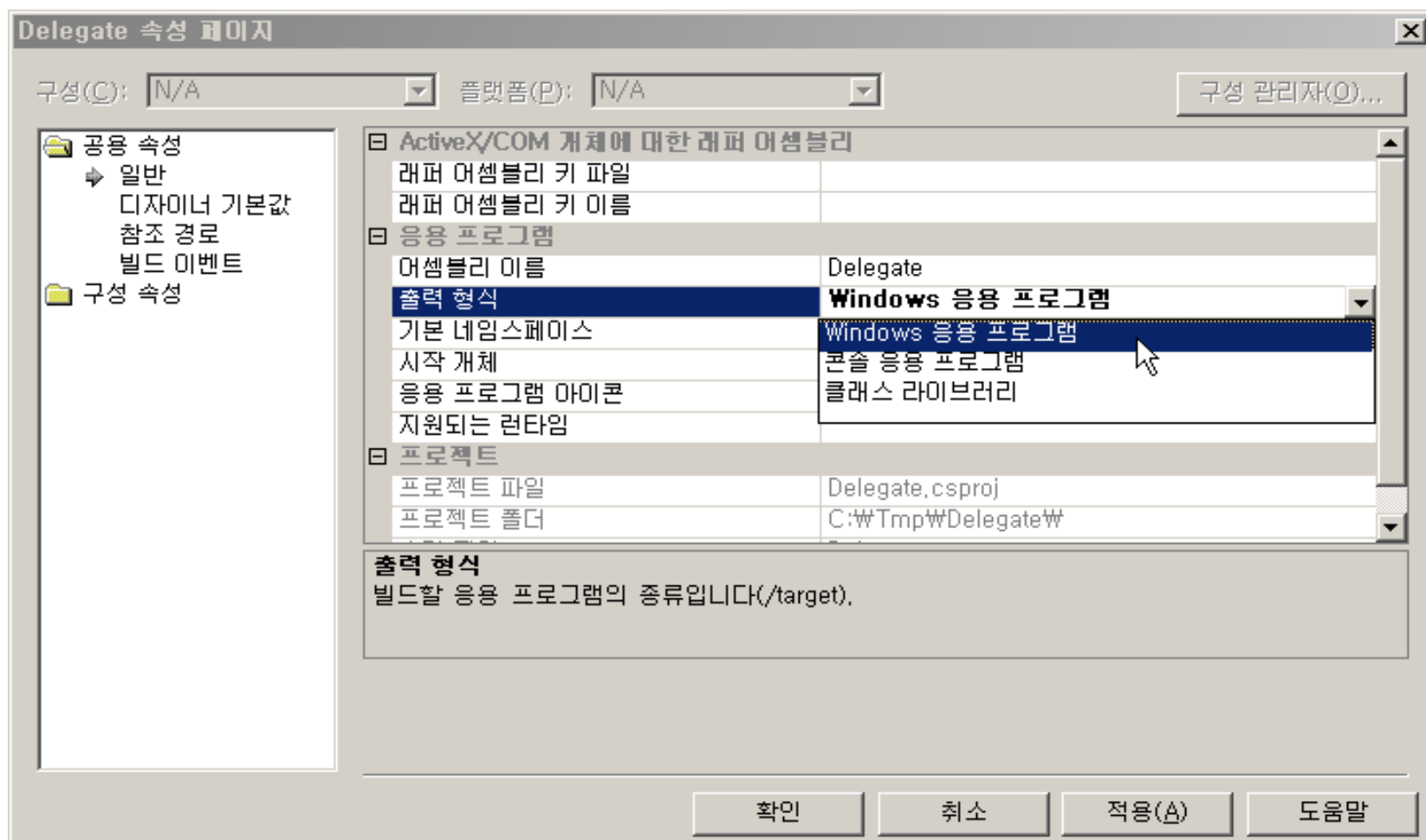
17. 이벤트 처리과정

- **프로그램을 실행하면**
 - 닷넷 런타임이 Main 함수 호출
 - Main 함수에서 Run 함수 호출
 - Run 함수에서 폼 객체(길동이) 얼굴(윈도우) 표시
 - 이후 Run 함수에서 호주머니(이벤트 큐)를 계속 확인 (무한루프)
- **여러분이 마우스를 클릭 (이벤트 발생)**
 - 운영체제(윈도)에서 Click 이벤트를 만들어 호주머니에 넣음.
 - 호주머니를 계속 체크하던 Run 함수에서 이벤트를 꺼내 확인
 - Click 이벤트임을 알고 길동이 OnClick 가상함수 호출
 - OnClick 가상함수 에서 Click 델리게이트가 null이 아니면 여러분이 작성한 핸들러 함수 xxx 호출
- **여러분이 윈도우를 닫을 경우 (이벤트 발생)**
 - Close 이벤트 발생
 - Run 함수에서 길동이 OnClose 가상함수 호출
 - 프로그램 종료

18. 윈도우 응용 설정하기

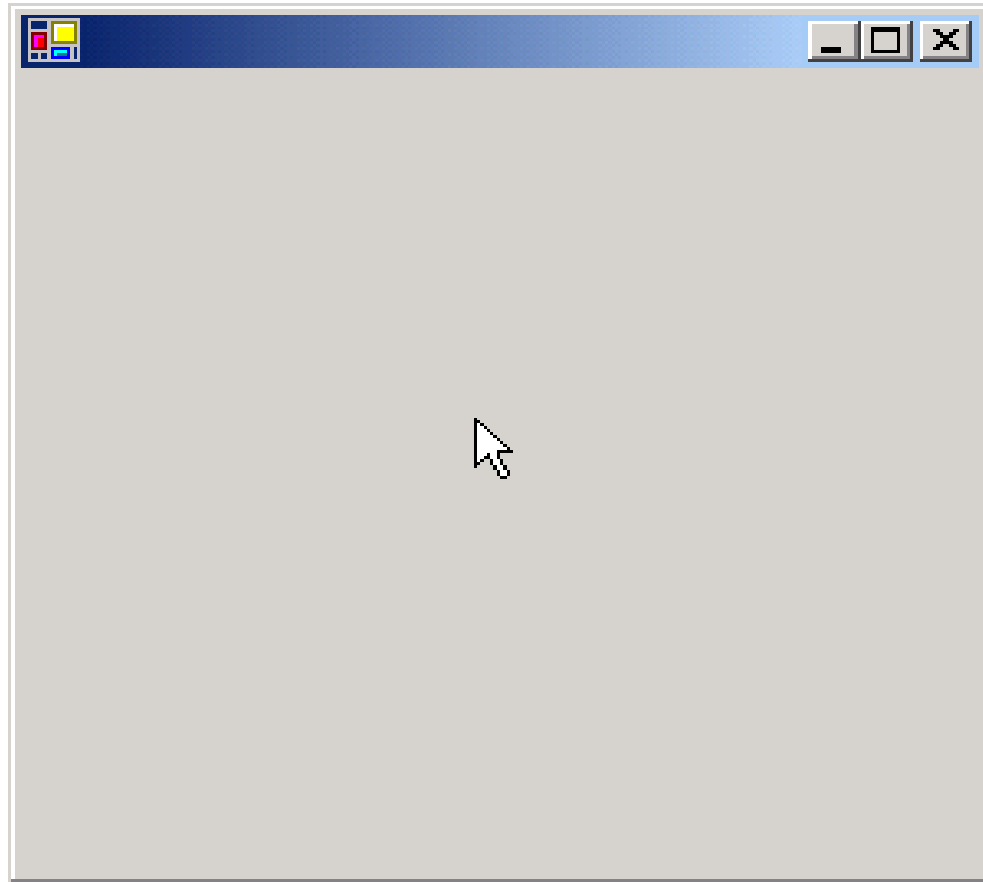


18. 윈도우 응용 설정하기



18. 윈도우 응용 설정하기

- 도스창은 표시되지 않고 **폼** 윈도우만 표시됨.



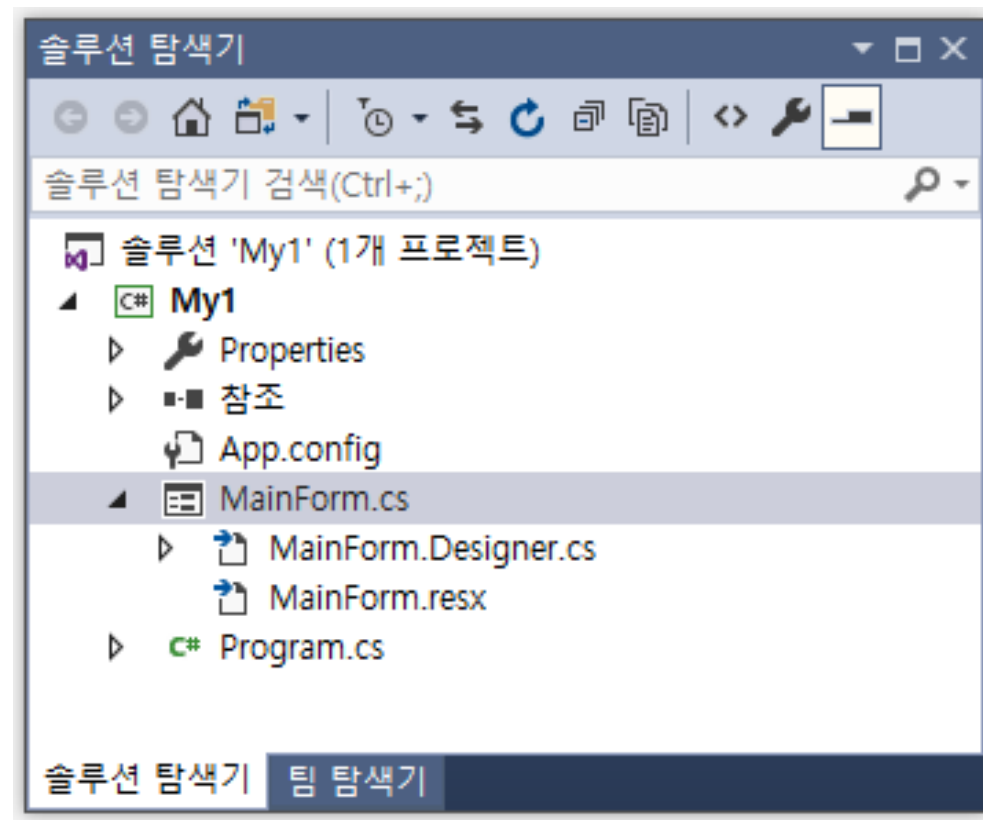
19. 코드 다듬기

- Derived 클래스 -> MainForm 클래스
- MainForm 클래스 생성자 함수 내의 코드를 InitializeComponent 함수로 추상화
- 네임 스페이스 XXX 지정

```
namespace XXX  
{  
  
}
```

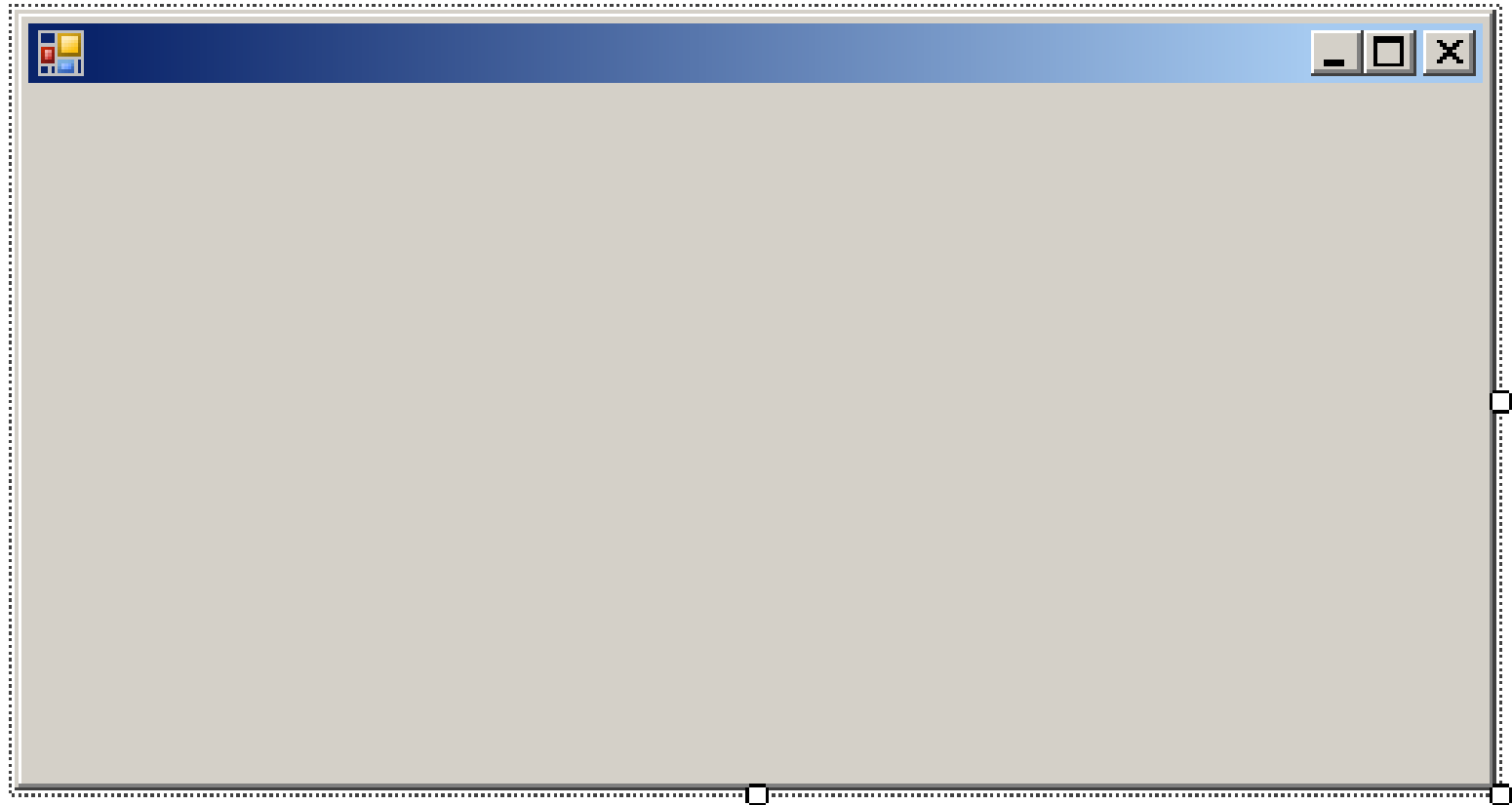
20. 디자인 편집기를 이용한 UI 디자인

- 비주얼 디자인 (MainForm.cs 두 번 클릭)



20. 디자인 편집기를 이용한 UI 디자인

- 폼 윈도우 크기 변경하기

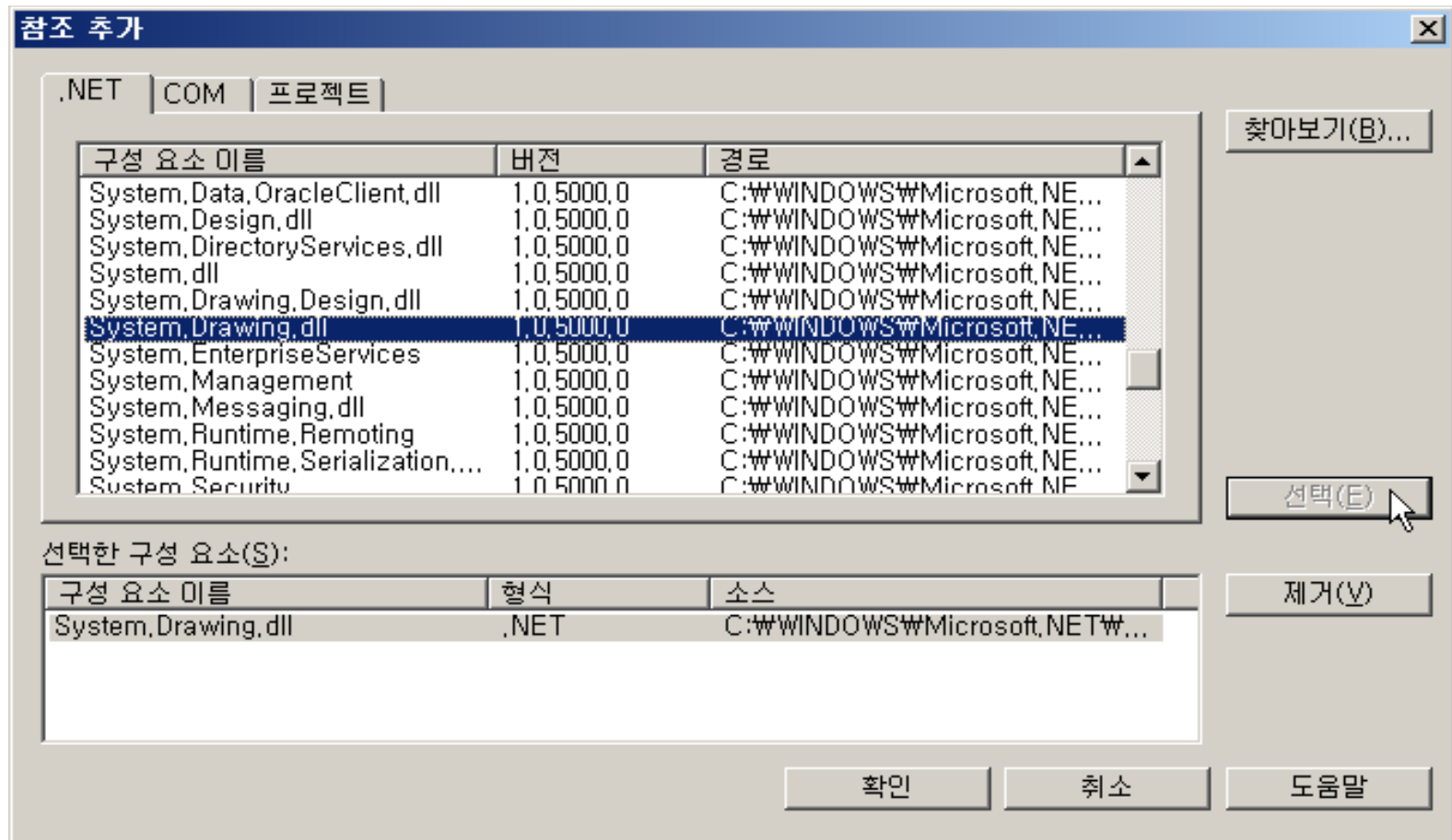


20. 디자인 편집기를 이용한 UI 디자인

```
private void InitializeComponent()
{
    this.SuspendLayout();
    //
    // MainForm
    //
    this.ClientSize = new System.Drawing.Size(440, 209);
    this.Name = "MainForm";
    this.Load += new
        System.EventHandler(this.MainForm_Load);
    this.Click += new System.EventHandler(this.xxx);
    this.ResumeLayout(false);
}
```

System.Drawing.dll
참조를 추가함

20. 디자인 편집기를 이용한 UI 디자인



20. 디자인 편집기를 이용한 UI 디자인

- 디자인 편집기를 이용하면
 - 코드가 **자동으로 생성**됨 → InitializeComponent 멤버 함수에
 - 결국 InitializeComponent 멤버 함수에 자동으로 작성되는 코드는 '**디자인 코드**'
 - 하지만 그 이외에는 디자인 코드가 아닌 **일반코드**임.
가령 이벤트 핸들러 함수인 xxx 함수

21. 디자인코드와 일반코드의 분리

- 또 다른 MainForm partial 클래스 작성
- InitializeComponent 멤버 함수를 그 클래스로 옮김

```
public partial class MainForm
{
    private void InitializeComponent()
    {
    }
}
```

21. 디자인코드와 일반코드의 분리

```
using System.Windows.Forms;

public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
    }
}
```

```
using System.Windows.Forms;

public class My
{
    static void Main()
    {
        Application.Run(new MainForm());
    }
}
```

22. 품응용프로젝트 작성

- 자동으로 작성 후 소스 분석해보자.