

제6장

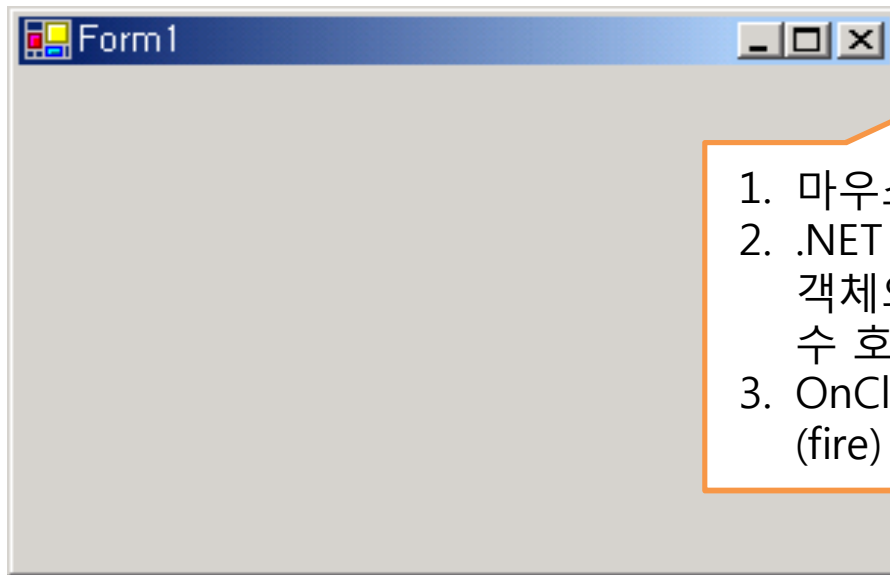
델리게이트와 닷넷 프레임워크

제주대학교 컴퓨터공학과
변영철 교수

<http://groups.google.com/group/jnu-cp2>

7. 델리게이트와 이벤트

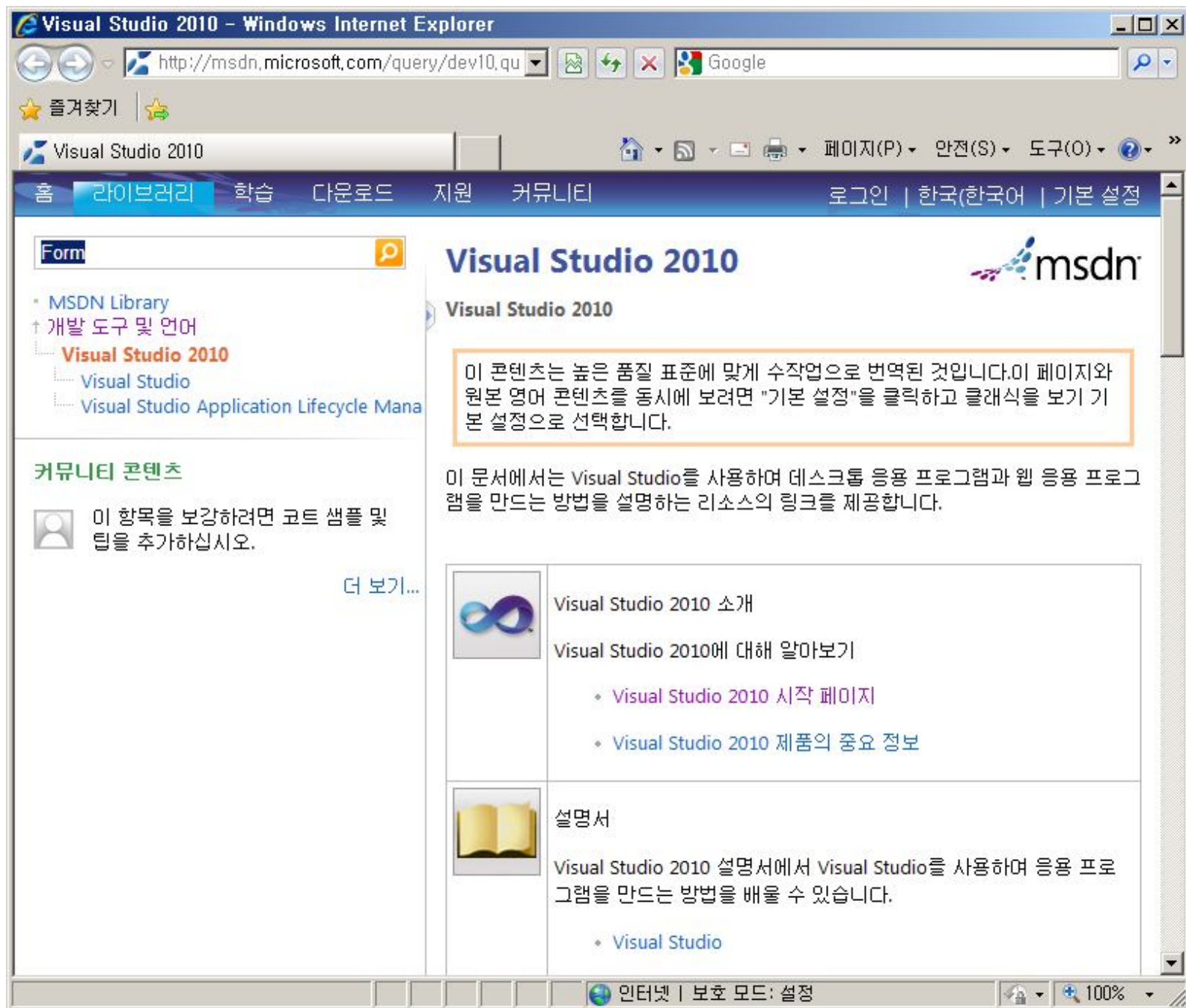
- 폼 윈도우와 이벤트

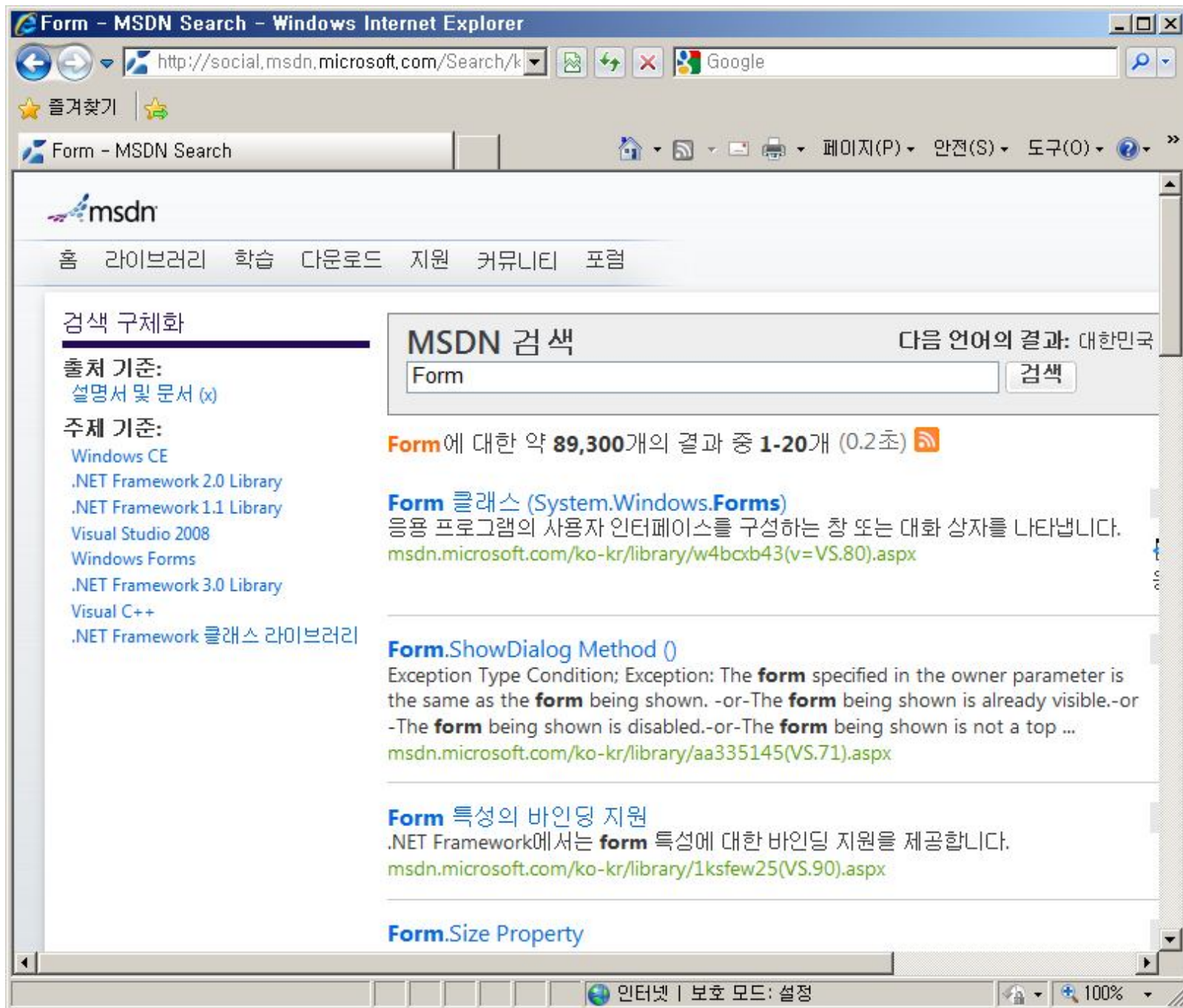


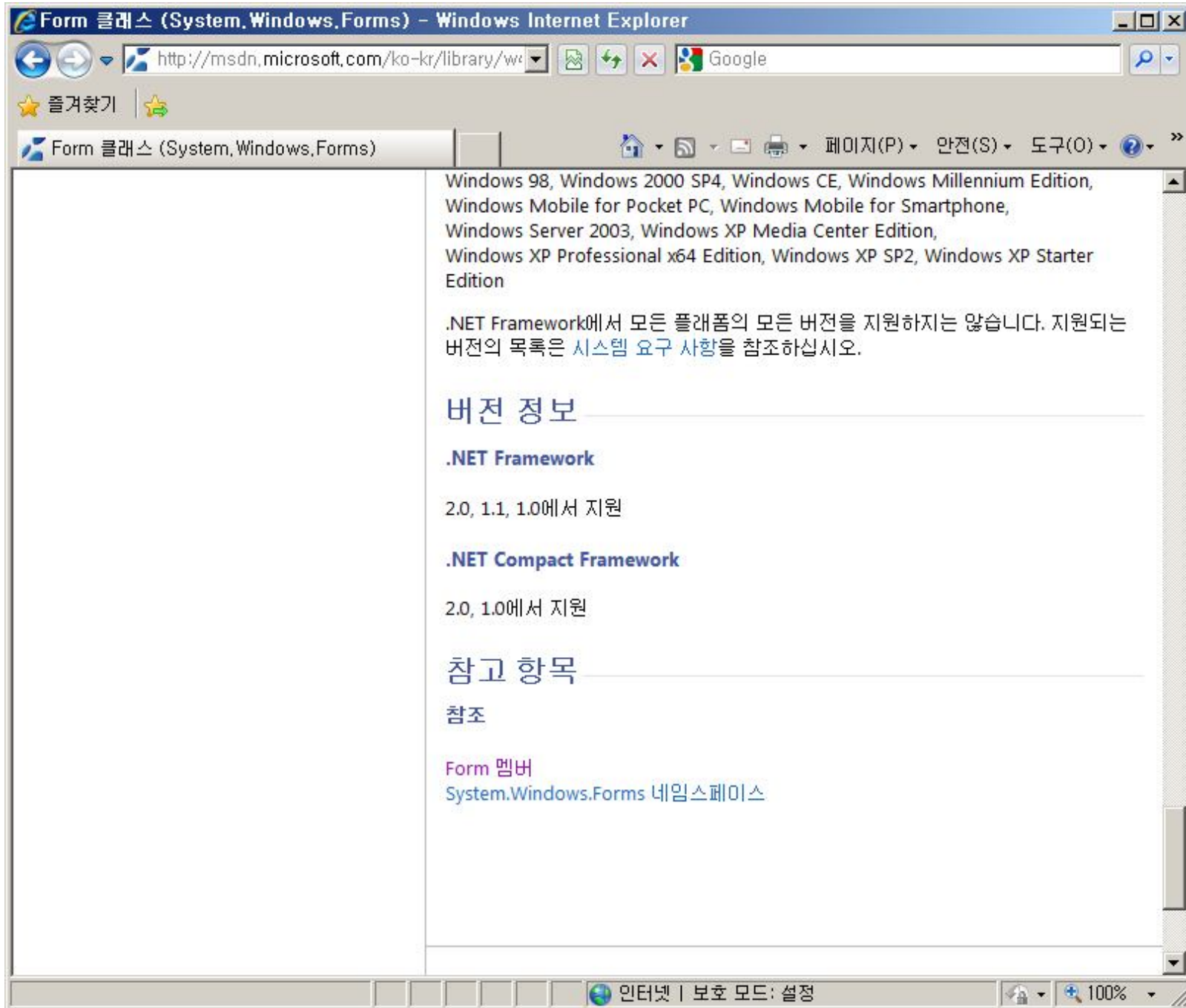
1. 마우스로 폼 윈도우 클릭
2. .NET 런타임(runtime)은 폼 윈도우 객체의 OnClick이라는 가상 멤버 함수 호출
3. OnClick 메소드는 Click 이벤트 발생 (fire)

- OnClick 도움말

 - 도움말 > 도움말 보기 메뉴 선택







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%

7. 델리게이트와 이벤트

- OnClick 함수의 모습

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


7. 델리게이트와 이벤트

- 흉내내기

```
using System;

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

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

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

    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 가상 함수 |

7. 델리게이트와 이벤트

- DelegateClass -> MyHandler
 - Click 이벤트는 핸들러 함수를 대신 호출하기 위해 만든 델리게이트 객체

7. 델리게이트와 이벤트

```
using System;
```

```
public class Base  
{
```

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

```
    public Base()  
    {  
        Click = new MyHandler(xxx);  
    }
```

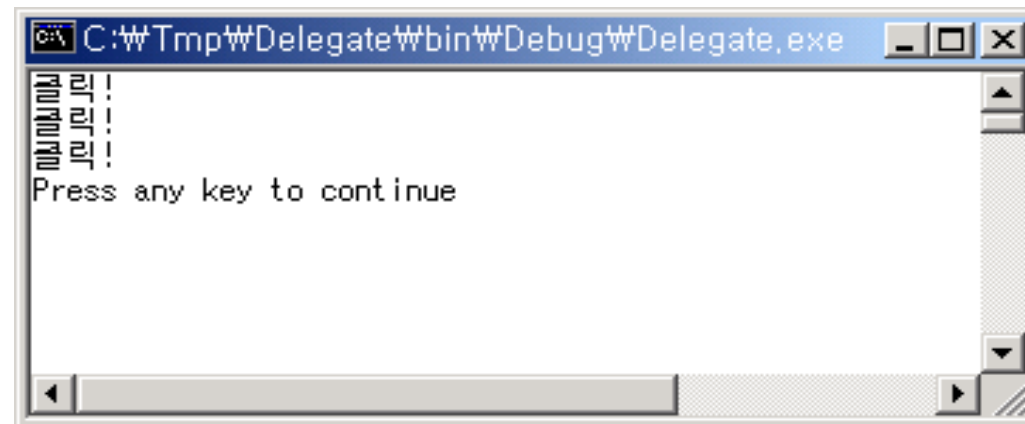
8. 델리게이트가 필요한 이유

- public delegate **void** MyHandler();
- 리턴값이 없고(void) 파라미터가 없는 모든 함수를 대신해서 사용 가능함
- 핸들러 함수 이름이 아직 결정되지 않아도 델리게이트로 호출할 수 있다는 뜻
- 핸들러 함수(xxx) 이름은 프로그래머가 결정하는 것
- 즉, 델리게이트가 있는 이유는 정해져 있지 않은 핸들러 함수를 호출하기 위한 것

8. 델리게이트가 필요한 이유

- Click 델리게이트에 함수를 여러 번 부착할 경우

```
public Base()
{
    Click += new MyHandler(xxx);
    Click += new MyHandler(xxx);
    Click += new MyHandler(xxx);
}
```



8. 델리게이트가 필요한 이유

- 두 개 이상의 함수 연결

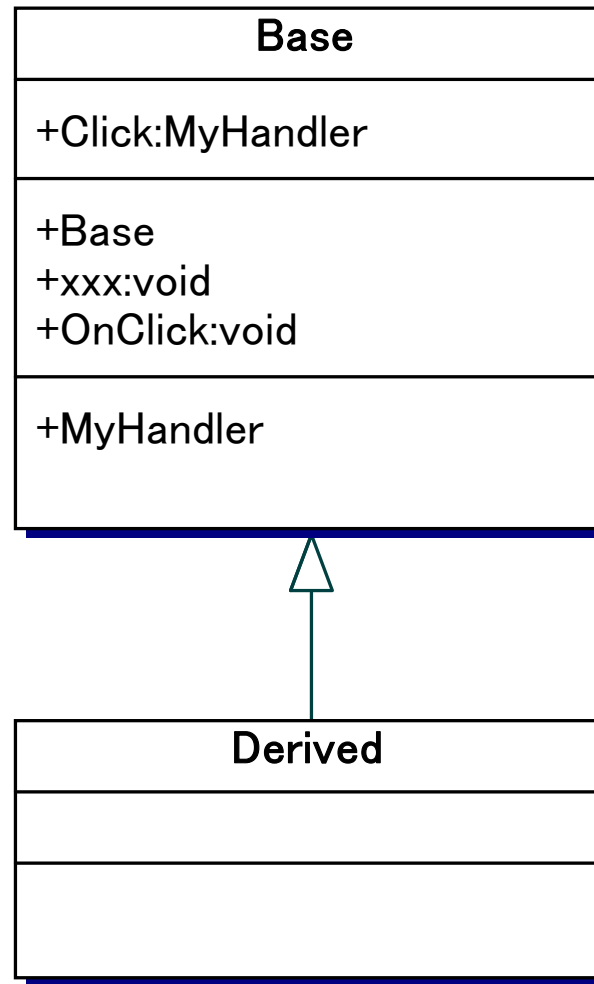
```
public Base()
{
    Click += new MyHandler(xxx);
    Click += new MyHandler(xxx);
    Click += new MyHandler(xxx);
    Click += new MyHandler(aaa);
}

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

9. 비하인드 코드, 비하인드 클래스

- Derived 클래스 작성
 - Derived : Base
 - Base 클래스 안에 작성된 코드가 고스란히 Derived 클래스 안으로 들어옴(재사용, 상속)
 - 따라서 Derived 클래스 객체 이용해도 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)

9. 비하인드 코드, 비하인드 클래스



10. Base 클래스 라이브러리화

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

10. Base 클래스 라이브러리화

- 문제점
 - xxx라는 핸들러 함수 이름은 프로그래머가 결정하는 것이므로 Base 클래스 안에 기술되어 있으면 Base는 라이브러리가 될 수 없음
- xxx 관련 코드를 프론트 클래스로 옮기기

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

10. Base 클래스 라이브러리화

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

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

11. 가상 함수를 이용한 이벤트 처리

- Base 클래스의 OnClick을 가상 함수로 만들기

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

11. 가상 함수를 이용한 이벤트 처리

- Derived 클래스에서 오버라이딩

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

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

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



11. 가상 함수를 이용한 이벤트 처리

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



11. 가상 함수를 이용한 이벤트 처리

- 베이스 클래스의 OnClick 호출하도록 작성

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

11. 가상 함수를 이용한 이벤트 처리

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

12. 코드 다듬기

- 클래스 이름 바꾸기
 - Base -> Form
 - MyHandler -> EventHandler

13. Application 응용 클래스 작성

- Main 함수 내의 코드를 Run으로 추상화

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

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

13. Application 응용 클래스 작성

- 객체 정의 및 호출

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

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


13. Application 응용 클래스 작성

- 정적 멤버로 선언

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

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

13. Application 응용 클래스 작성

- 독립된 클래스로 작성

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

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

14. Application 클래스 라이브러리화

- 문제점
 - 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

14. Application 클래스 라이브러리화

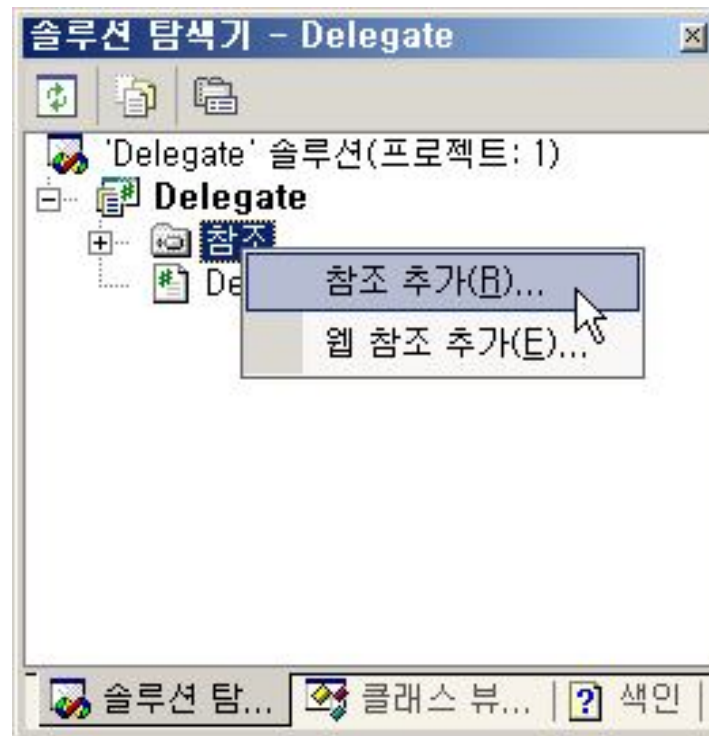
- 라이브러리화 가능하게 하기

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

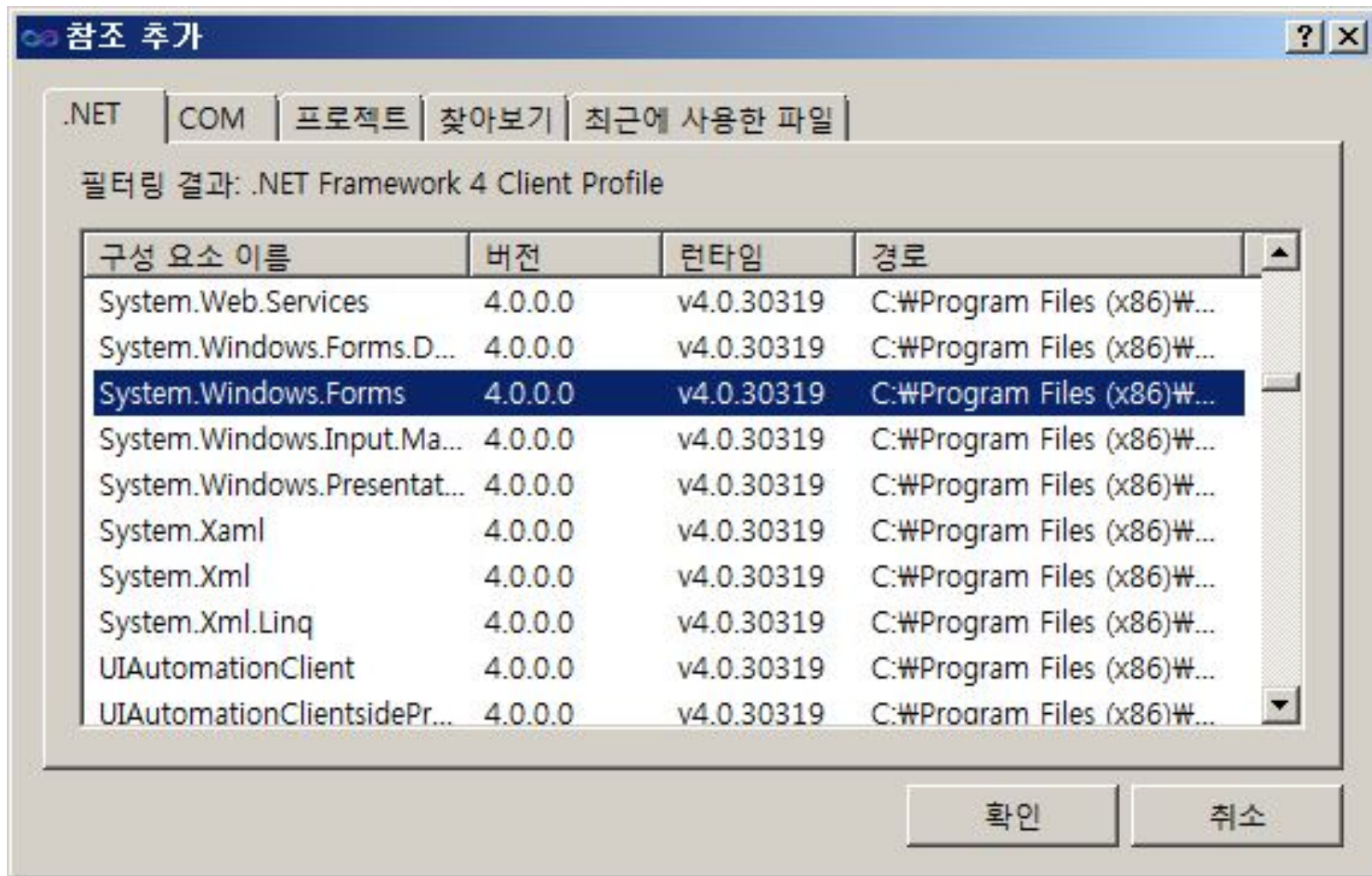
public class Delegate
{
    public static void Main()
    {
        Application.Run(new Derived());
    }
}
```

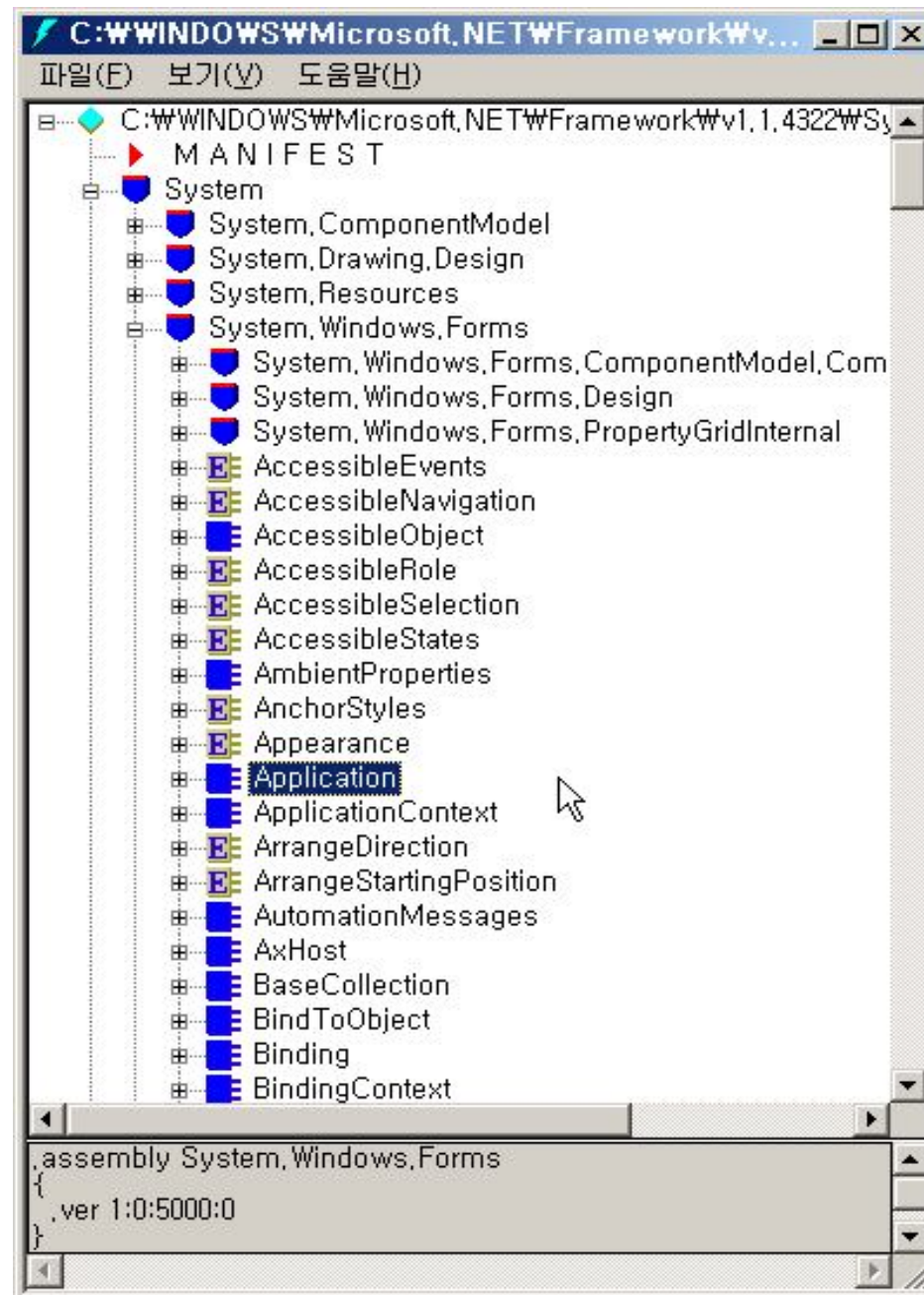
15. C# 클래스 라이브러리 이용하기

- Form, Application 클래스는 닷넷 라이브러리에 존재함
- 따라서 힘들게 작성하기 보다는 이를 이용할 수 있음.



15. C# 클래스 라이브러리 이용하기





15. C# 클래스 라이브러리 이용하기

- 두 클래스 제거하기
- 다음 코드 추가하기

`using System.Windows.Forms;`

- EventHandler 선언 고치기

`public delegate void EventHandler(Object o,
EventArgs e);`

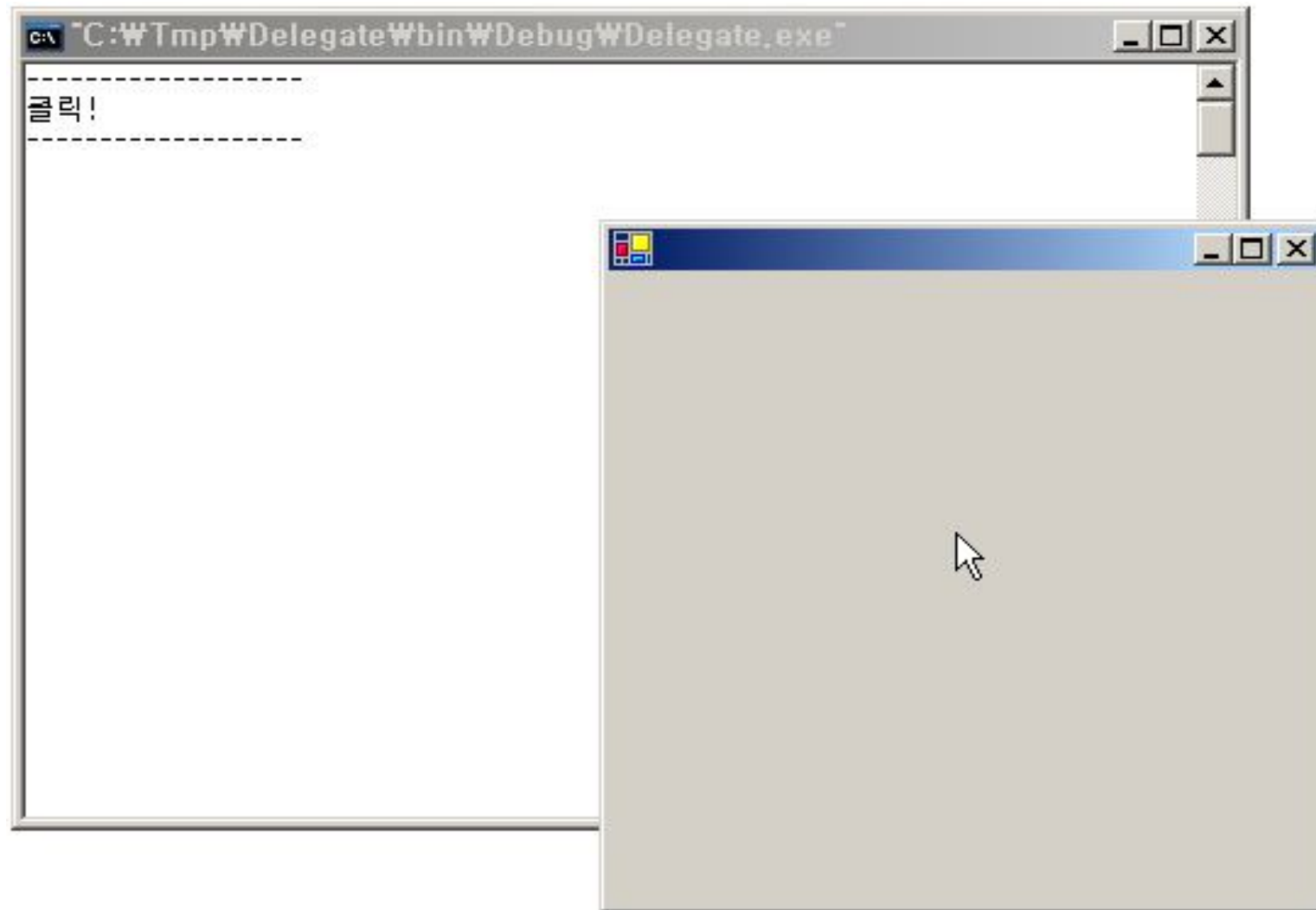
15. C# 클래스 라이브러리 이용하기

- 핸들러 함수 xxx 수정

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

- OnClick 가상함수 수정

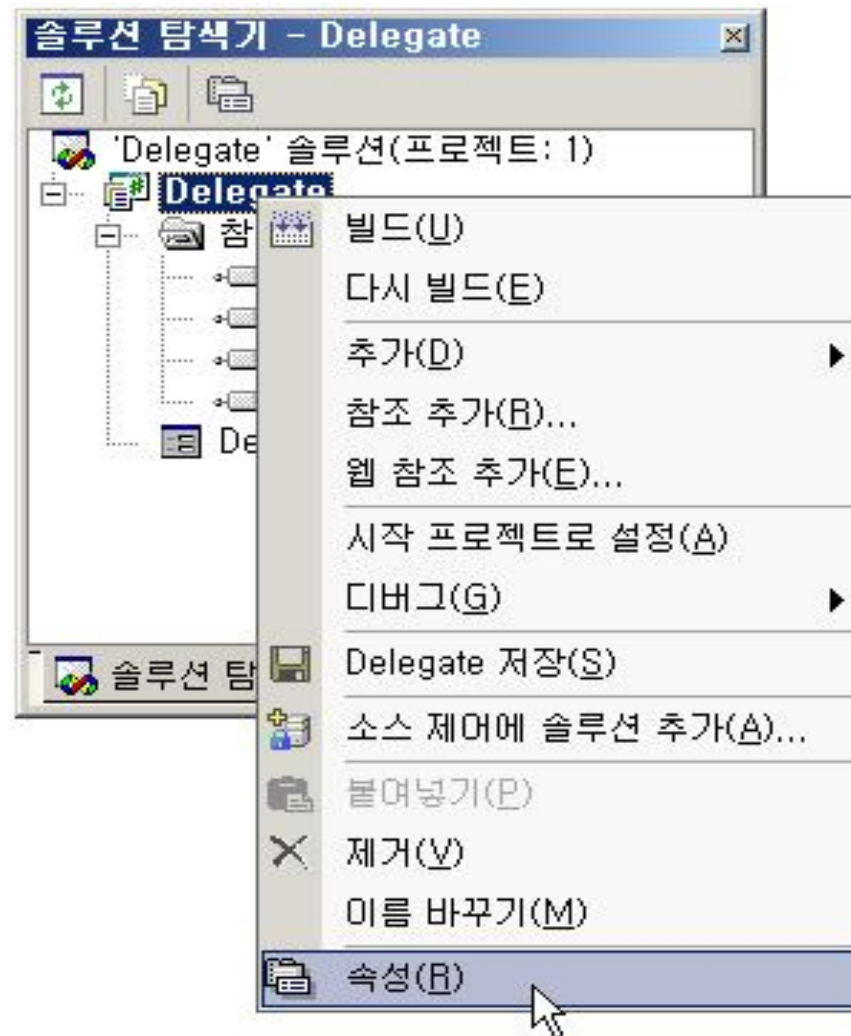
```
protected override void OnClick(EventArgs e)
{
    Console.WriteLine("-----");
    base.OnClick(e);
    Console.WriteLine("-----");
}
```



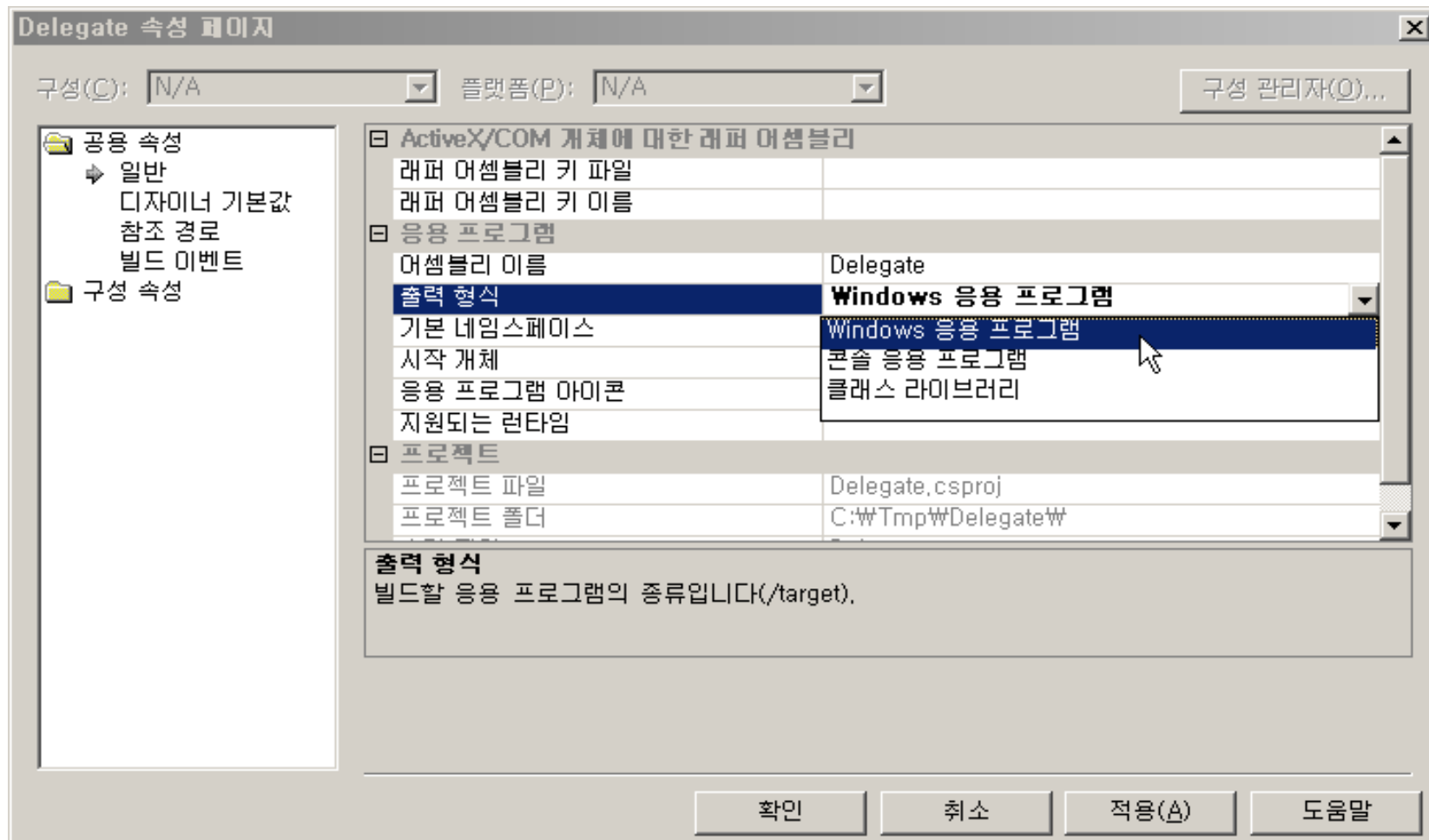
16. 이벤트 처리과정

- 프로그램을 실행하면
 1. 닷넷 런타임은 Main 함수를 호출
 2. Main 함수에서 Run 함수 호출
 - (1) 폼 객체의 얼굴에 해당하는 윈도우를 표시
 - (2) 여러분이 마우스를 클릭
 - (3) Click 이벤트를 발생
 - (4) 내부적으로 OnClick 가상 함수(여러분이 재정의한 함수)를 자동으로 호출
 - (5) 윈도우를 닫을 경우 Run 함수도 종료

17. 윈도우 응용 설정하기

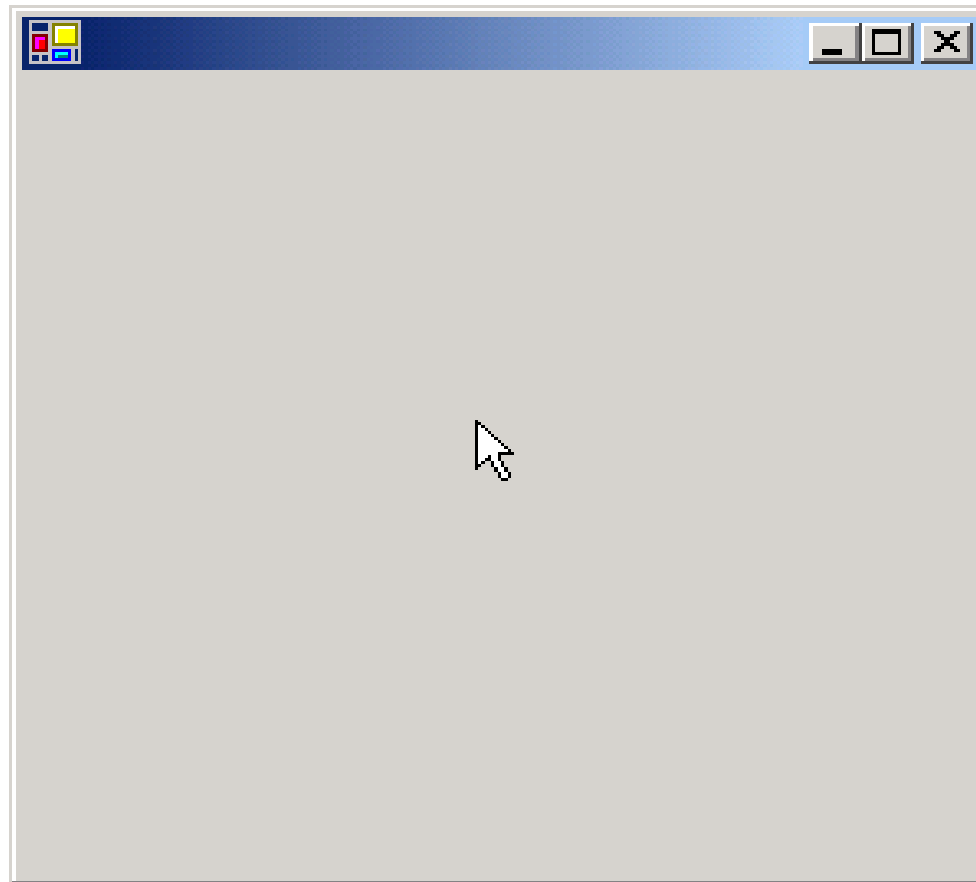


17. 윈도우 응용 설정하기



17. 윈도우 응용 설정하기

- 도스창은 표시되지 않음



18. 코드 다듬기

- Derived 클래스 -> MainForm 클래스
- MainForm 클래스 생성자 함수 내의 코드를 InitializeComponent로 추상화
- 네임 스페이스 지정

```
namespace XXX  
{  
  
}
```