

Microsoft Windows Mobile

Dae-Ki Kang

차례

- 소개 (Introduction)
- 설치 (Installation)
- 헬로 (Hello, Smartphone! etc. in MFC of C++)
- 배포 (Deployment (upload a program to a smart device))
- GUI (GUI Programming in C#)
- C#의 개요
- **C# 프로그래밍 실습 (More Serious Issues in C#)**

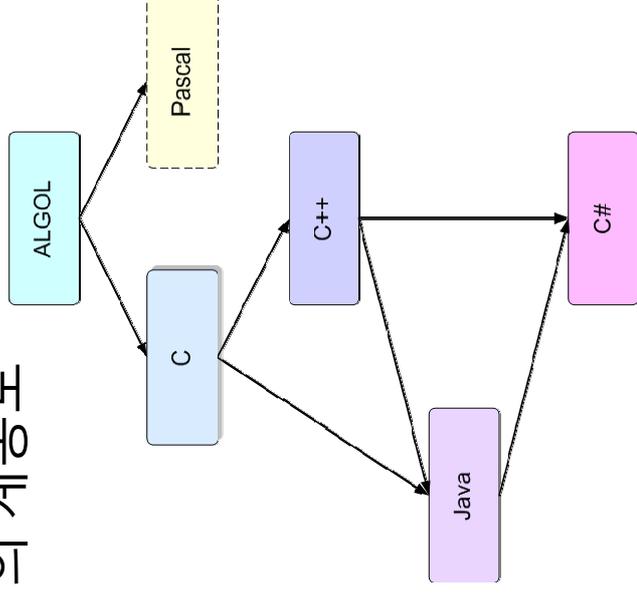
소개 [1/2]

- C# 프로그래밍 언어
 - MS사의 앤더스 헬스버그(Anders Hejlsberg)가 고안
 - .NET에 최적화된 언어
 - 컴포넌트 지향 프로그래밍 언어
 - 자바의 단점을 보완
 - 실행 방법: 자바: 인터프리케이션, C#: 컴파일 방법
 - 자바 언어를 대체할 수 있는 언어
- C#의 특징
 - 객체지향 언어: 자료 추상화
 - 델리게이트와 이벤트
 - 멀티스레드, 예외처리
 - 연산자 중복, 제네릭

소개 [2/2]

- C 계열의 언어
 - C++와 자바로부터 영향을 받았음.
 - C: 연산자와 문장 등 기본적인 언어의 기능
 - C++: 객체지향 속성, 연산자 중복, 제네릭(Generic)
 - 자바: 예외처리와 스레드

■ C# 언어의 계통도



콘솔/윈폼 애플리케이션

- 콘솔 애플리케이션
 - 문자기반 명령어 프롬프트 환경에서 실행
 - 키보드를 통해 입력, 화면에 문자로 출력
- 윈폼 애플리케이션
 - 윈도우 폼 애플리케이션(Windows forms Application)의 약어
 - GUI를 통해서 입출력을 수행
 - 이벤트 처리 방식을 통해 실행
- C# 개발 환경
 - SDK를 이용 - 편집기, 컴파일러, 실행엔진, 클래스 라이브러리
 - 통합개발 환경(IDE)
 - Visual Studio .NET

콘솔 애플리케이션

- 일반적인 응용 프로그램
- 예제 프로그램:

[예제 1.1 – HelloWorld.cs]

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

실행 결과 :
Hello World!

네임스페이스

출력 메소드

- 실행 과정

```
C:\temp>csc HelloWorld.cs
C:\temp>HelloWorld
Hello World!
```

윈폼 애플리케이션

- 윈도우 응용 프로그램
- 예제 프로그램:

[예제 1.2 – WinFormApp.cs]

```
using System;
using System.Windows.Forms;
class WinFormApp : Form {
    WinFormApp() {
        Text = "Hello World!";
    }
    public static void Main() {
        Application.Run(new WinFormApp());
    }
}
```

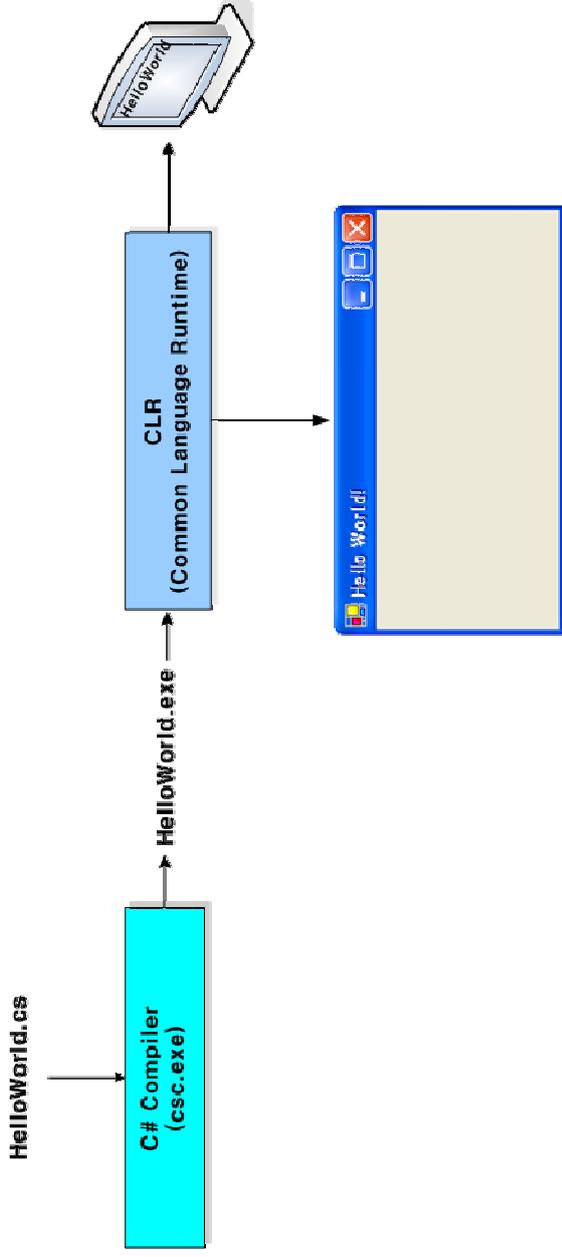
실행 결과 :



■ Text 프로퍼티
■ 타이틀 바에 출력

C# 프로그램 실행과정

- 컴파일 과정
 - `csc : C# compiler`
- 실행 시스템
 - CLR – Common Language Runtime
- 실행 과정



기본 특징 [1/4]

- 자료형
 - 변수나 상수가 가질 수 있는 값과 연산의 종류를 결정
 - C#의 자료형
 - 값형(value type)
 - 참조형(reference type)
 - 숫자형
 - 정수형
 - signed – sbyte, short, int, long
 - unsigned – byte, ushort, uint, ulong
 - 실수형 – float, double, decimal
- 연산자
 - 표준 C 언어와 유사
 - 형 검사 연산자(type testing operator)
 - is – 호환 가능한지를 검사
 - as – 지정한 형으로 변환

기본 특징 [2/4]

- 배열형
 - 같은 형을 갖는 여러 개의 값을 저장할 수 있는 자료구조.
- 배열 변수 선언
 - 배열을 가리키는 참조 변수

```
int[] vector;  
short[,] matrix;  
long[][] arrayOfArray;  
object[] myArray1, myArray2;
```

기본 특징 [3/4]

- 배열 객체 생성
- new 연산자

```
vector = new int[100];  
matrix = new short[10, 100];  
myArray1 = new Point[3];
```

- 배열 사용

```
for (int i=0; i<vector.Length; i++)  
    vector[i] = i;
```

기본 특징 [4/4]

- **스트링**
 - C#에서 스트링은 객체
 - System.String 클래스의 객체
 - C#의 string 형은 String 클래스의 alias
- **스트링 상수**
 - 이중 인용부호(“)로 묶인 문자들의 나열 (예 : “I am a string.”)
- **스트링 초기화**

```
string s = "Hello";  
string s = new string("Hello");
```

- **스트링 연결**
 - + 연산자 : concatenation operator

```
string s = "Hello";  
s += " World";  
=> s: Hello World
```

주요 특징

- 클래스
- 프로퍼티
- 연산자 중복
- 델리게이트
- 이벤트
- 스레드
- 제네릭

클래스

- 클래스와 객체
- 프로그래밍 언어적인 측면에서
 - 객체 자료형 또는 객체 클래스
- 클래스의 구성 - 클래스 멤버
 - 필드 계통 - 상수 정의, 필드, 이벤트
 - 메소드 계통 - 메소드, 생성자, 소멸자, 프로퍼티, 인덱서, 연산자 중복

자동차 출발!!!



클래스의 설계

- 객체의 속성과 행위를 결정
 - 속성 - 필드 계통
 - 행위 - 메소드 계통

```
class CoffeeMaker {  
    public bool onState;  
    public void StartCoffeeMaker() {  
        if (onState == true)  
            Console.WriteLine("The CoffeMaker is already on");  
        else  
            onState = true;  
        Console.WriteLine("The CoffeMaker is now on");  
    }  
}
```

 CoffeeMaker 클래스를 이용하여 StartCoffeeMaker() 메소드를 호출하여 보시오.

프로퍼티 [1/2]

- 프로퍼티의 개념
 - 클래스의 private 필드를 형식적으로 다루는 일종의 메소드.
 - 값을 지정하는 셋-접근자와 값을 참조하는 갓-접근자로 구성.
- 프로퍼티의 정의 형태

```
[property-modifiers] returnType PropertyName {  
    get {  
        // get-accessor body  
    }  
    set {  
        // set-accessor body  
    }  
}
```

프로퍼티 [2/2]

- 프로퍼티의 동작
 - 필드처럼 사용되지만, 메소드처럼 동작.
 - 배경문의 왼쪽에서 사용되면 셋-접근자 호출.
 - 배경문의 오른쪽에서 사용되면 갯-접근자 호출.
- 예제 1.5
 - Text p. 28
 - Value 지정어의 의미 ?

연산자 중복

- 연산자 중복의 의미
 - 시스템에서 제공한 연산자를 재정의 하는 것.
 - 클래스만을 위한 연산자로서 자료 추상화가 가능.
 - 시스템에서 제공한 연산자처럼 사용 가능
 - 문법적인 규칙은 변경 불가(연산 순위나 결합 법칙 등).

- 연산자 중복 정의 형태

```
public static [extern] returnType operator op (parameter1 [, parameter2]) {  
    // ... operator overloading body ...  
}
```

- 예제 1.6 – 교과서 29쪽

델리게이트

- 델리게이트(delegate)는 메소드 참조 기법
 - 객체지향적 특징이 반영된 메소드 포인터
- 이벤트와 스레드를 처리하기 위한 방법론
- 선언 형태:

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

- 델리게이트 프로그래밍 순서
 - ① 메소드 정의
 - ② 델리게이트 정의
 - 델리게이트 형태와 연결될 메소드의 형태가 일치
 - 반환형, 매개변수의 개수와 형
 - ③ 델리게이트 객체 생성
 - ④ 델리게이트 객체에 메소드 연결
 - ⑤ 델리게이트를 통해 메소드 호출

이벤트 [1/2]

- 이벤트(event)
 - 사용자 행동에 의해 발생하는 사건.
 - 어떤 사건이 발생한 것을 알리기 위해 보내는 메시지로 간주.
 - C#에서는 델리게이트를 이용하여 이벤트를 처리.
 - 이벤트 정의 형태
- ```
[event-modifier] event DelegateType EventName;
```
- 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

# 이벤트 [2/2]

- 이벤트 프로그래밍 순서
  - ① 이벤트 처리기를 작성
  - ② 이벤트 처리기의 형태와 일치하는 델리게이트를 정의  
(또는 System.EventHandler 델리게이트를 사용)
  - ③ 델리게이트를 이용하여 이벤트를 선언  
(미리 정의된 이벤트인 경우에는 생략)
  - ④ 이벤트에 이벤트 처리기를 등록
  - ⑤ 이벤트를 발생  
(미리 정의된 이벤트는 사용자 행동에 의해 이벤트가 발생)
- 이벤트가 발생되면 등록된 메소드가 호출되어 이벤트를 처리
  - 미리 정의된 이벤트 발생은 사용자의 행동에 의해서 발생
  - 사용자 정의 이벤트인 경우에는 명시적으로 델리게이트 객체를 호출함으로써 이벤트 처리기를 작동

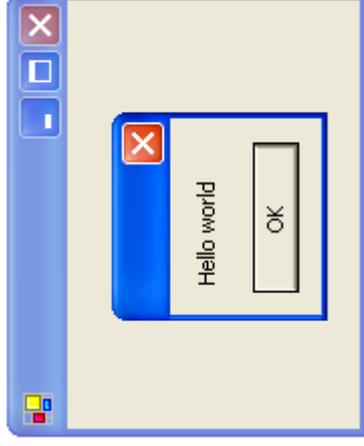
# [예제 1.8] EventApp.cs

[예제] 1.8 - EventApp.cs]

```
using System;
using System.Windows.Forms;
class EventApp : Form {
public EventApp() {
 this.Click += new EventHandler(ClickEvent); // ④ 이벤트 처리기 등록
}
void ClickEvent(object, sender, EventArgs) { // ① 이벤트 처리기 작성
 MessageBox.Show("Hello world");
}
public static void Main() {
 Application.Run(new EventApp());
}
}
```

미리 정의된 이벤트를 위한  
델리게이트

실행 결과 :



# 스레드

- 스레드의 기본 개념
  - 순차 프로그램과 유사하게 시작, 실행, 종료의 순서를 가짐
  - 실행되는 동안에 한 시점에서 단일 실행 점을 가짐
  - 프로그램 내에서만 실행 가능
    - 스레드는 프로그램 내부에 있는 제어의 단일 순차 흐름 (single sequential flow of control)
      - 단일 스레드 개념은 순차 프로그램과 유사
- 멀티 스레드 시스템
  - 스레드가 하나의 프로그램 내에 여러 개 존재
- 스레드 프로그래밍의 순서
  - ① 스레드 몸체에 해당하는 메소드를 작성
  - ② 작성된 메소드를 ThreadStart 델리게이트에 연결
  - ③ 생성된 델리게이트를 이용하여 스레드 객체를 생성
  - ④ 스레드의 실행을 시작(Start()) 메소드를 호출

# [예제 1.9] ThreadApp.cs

[예제 1.9 – ThreadApp.cs]

```
using System;
using System.Threading;
class ThreadApp {
 static void ThreadBody() {
 Console.WriteLine("In the thread body ...");
 }
 public static void Main() {
 ThreadStart ts = new ThreadStart(ThreadBody);
 Thread t = new Thread(ts);
 Console.WriteLine("*** Start of Main");
 t.Start();
 Console.WriteLine("*** End of Main");
 }
}
```

// 반드시 포함 !!!  
// --- ①  
// --- ②  
// --- ③  
// --- ④

실행 결과 :

```
*** Start of Main
*** End of Main
In the thread body ...
```

# 제네릭

- 제네릭의 의미
  - 자료형을 매개변수로 가질 수 있는 개념
  - C++의 템플릿과 유사한 개념
- 제네릭 단위
  - 클래스, 구조체, 인터페이스, 메소드
- 제네릭 클래스
  - 범용 클래스 또는 포괄 클래스
  - 형 매개변수(type parameter) - <> 안에 기술

---

```
예 : class Stack<StackType> {
 private StackType[] stack = new StackType[100];
 // ...
 public void Push(StackType element) { /* ... */ }
 public StackType Pop() { /* ... */ }
}
```

---

```
Stack<int> stk1 = new Stack<int>(); // 정수형 스택
Stack<double> stk2 = new Stack<double>(); // 실수형 스택
```

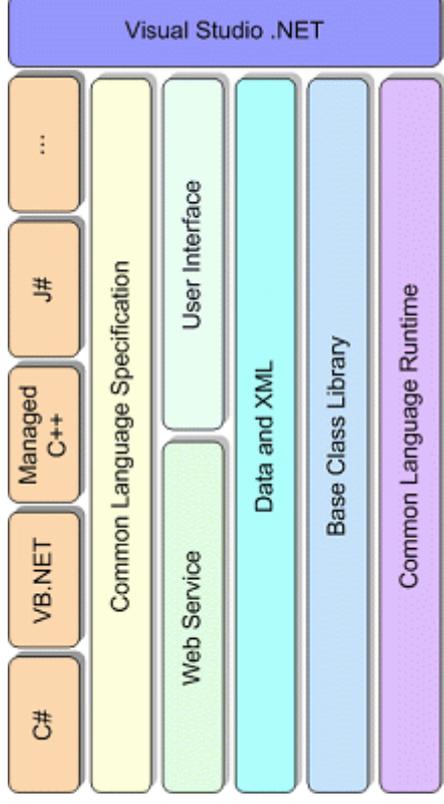
---

# .NET 프레임워크 소개

- 공통 언어 스펙
  - CLS: Common Language Specification
- 공통 자료형 시스템
  - CTS: Common Type System
- 실행 모델
  - 공통 언어 런타임
    - CLR: Common Language Runtime

# .NET 프레임워크

- .NET 프레임워크(Framework)
  - 마이크로소프트사가 개발한 프로그램 개발 환경.
- .NET 프레임워크의 구조도



- 현재 지원 언어
  - C#, Visual Basic .NET, Managed C++, J#
  - CLS 만족하는 언어

# 공통 언어 스펙

- 언어의 상호 운용성을 지원하기 위한 스펙
- CLS
  - Common Language Specification
  - CLS를 만족하는 언어를 "공통 언어"라 부름
  - C#, Visual Basic .NET, J#, Managed C++
- CLS를 만족하면 서로 다른 언어에서 만들어진 라이브러리를 공유할 수 있다.
- 기본 클래스 라이브러리
  - .NET 프레임워크에서 제공
  - BCL: Base Class Library
  - CLS를 만족하면 BCL을 사용할 수 있다.

# 공통 자료형 시스템

- 다른 언어와 상호 운용성에 필요한 공통의 자료형
- CTS 형과 C# 자료형 관계

| CTS 자료형        | 의미                    | C# 자료형  |
|----------------|-----------------------|---------|
| System.Object  | 객체형                   | object  |
| System.String  | 스트링형                  | string  |
| System.Sbyte   | 부호있는 바이트형(-128 ~ 127) | sbyte   |
| System.Byte    | 바이트형(0 ~ 255)         | byte    |
| System.Int16   | 16비트 정수형              | short   |
| System.UInt16  | 16비트 부호없는 정수형         | ushort  |
| System.Int32   | 32비트 정수형              | int     |
| System.UInt32  | 32비트 부호없는 정수형         | uint    |
| System.Int64   | 64비트 정수형              | long    |
| System.UInt64  | 64비트 부호없는 정수형         | ulong   |
| System.Char    | 문자형                   | char    |
| System.Single  | 단일 정밀도 실수형            | float   |
| System.Double  | 이중 정밀도 실수형            | double  |
| System.Boolean | 부울형                   | bool    |
| System.Decimal | 10진수형                 | decimal |

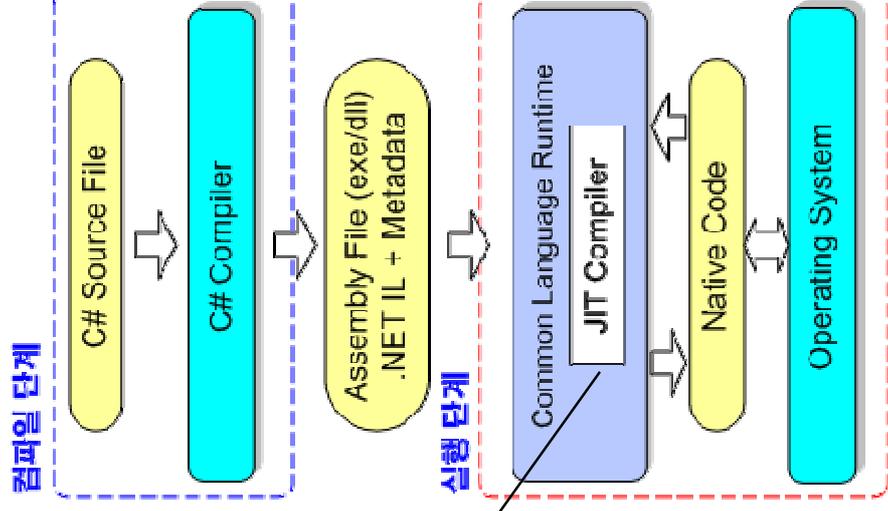
# 실행 모델

- 컴파일 단계
- C# Compiler

- 실행 단계

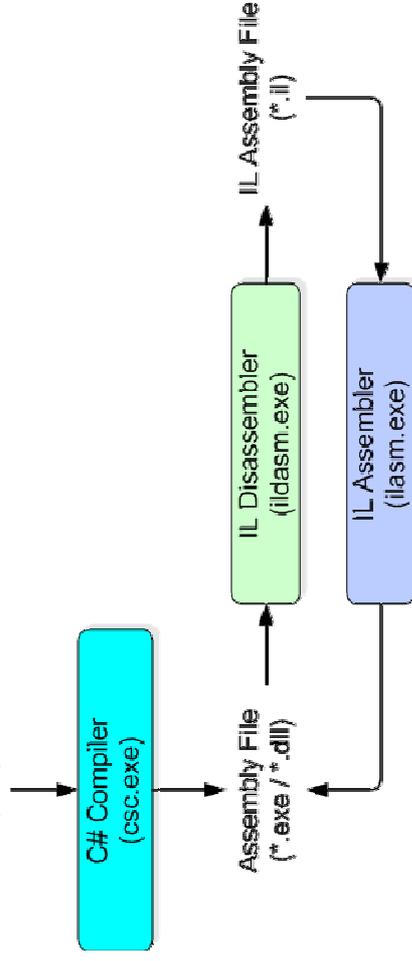
- Common Language Runtime

JIT 방법에 의해 실행



# 어셈블리와 역어셈블리 과정

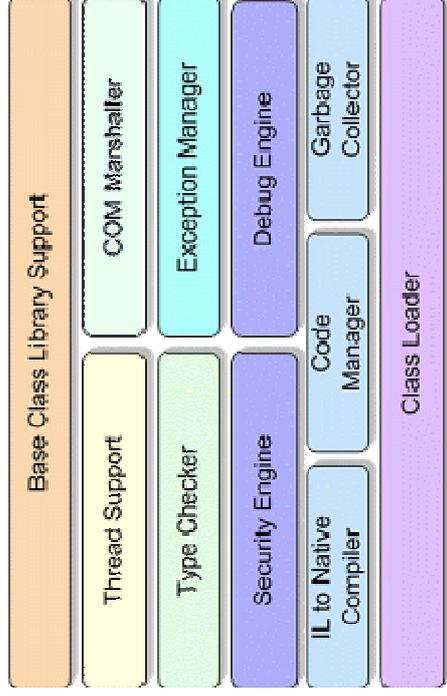
- 어셈블리 과정
  - \*.il → \*.exe or \*.dll
- 역어셈블리 과정
  - \*.exe or \*.dll → \*.il  
C# Source File (\*.cs)



- IL 파일
  - 텍스트 형태로 된 중간 언어 파일
  - 컴파일된 코드를 확인할 수 있음
  - 더 나아가 디버깅하는데 사용할 수 있음.

# 공통 언어 런타임

- .NET 프레임워크의 실행 시스템
- 자바의 JVM과 동일한 기능을 담당
- 실행 환경을 포함
  - 필수적인 실행 환경 3가지 컴포넌트
    - 메모리 관리기
    - 예외 처리기
    - 스레드 지원
- 공통 언어 런타임 컴포넌트



## 외부 링크들

- .NET Framework
  - [http://en.wikipedia.org/wiki/.NET\\_Framework](http://en.wikipedia.org/wiki/.NET_Framework)
- List of CLI languages
  - [http://en.wikipedia.org/wiki/List of .NET languages](http://en.wikipedia.org/wiki/List_of_.NET_languages)

# 차례

- 소개 (Introduction)
- 설치 (Installation)
- 헬로 (Hello, Smartphone! etc. in MFC of C++)
- 배포 (Deployment (upload a program to a smart device))
- GUI (GUI Programming in C#)
- C#의 개요
- **C# 프로그래밍 실습 (More Serious Issues in C#)**

# 파일 읽기 (Files)

- 파일을 읽어서 첫번째 라인을 레이블로 보인다

```
1. public Form1()
2. {
3. InitializeComponent();
4. ReadFileToLabel();
5. }

6. private void ReadFileToLabel()
7. {
8. StreamReader sr =
9. File.OpenText(@"\windows\pocketmsn.htm");
10. string s = "";
11. s = sr.ReadLine();
12. this.label1.Text = s;
}
```



# 데이터베이스 (Databases)

- 응용프로그램의 데이터와 세팅을 .NET 컴팩트 프레임워크를 이용하여 윈도우 모바일 기반의 스마트폰 응용 프로그램에 저장
- <http://msdn.microsoft.com/en-us/library/ms839432.aspx>
- XML 포맷의 데이터
  - XML Schema
  - XML files
  - DataSet
  - SqlCommand
- 근 미래 - LINQ 기술

## 프로세스 (Processes)

- Visual Studio.NET 2008 and .NET Compact Framework 2.0 or higher

```
1. private void menuItem1_Click(object sender,
EventArgs e)
2. {
3. Process.Start(@"IExplore.exe",
4. @"\"windows\"pocketmsn.htm");
}
```

속성

Form1 System, Windows.Forms, Form

|                          |                                      |
|--------------------------|--------------------------------------|
| 키 미리보기                   | False                                |
| 키 데이터 (DataBindings)     |                                      |
| 태그                       |                                      |
| 동작                       |                                      |
| AutoValidate Enabled     | True                                 |
| EnablePreventFocusChange | True                                 |
| 디자인                      |                                      |
| (Name)                   | Form1                                |
| FormFactor               | Windows Mobile 5.0 Smartphone (OVGA) |
| Language (기본값)           |                                      |
| Localizable              | False                                |
| Locked                   | True                                 |
| Skin                     | True                                 |
| 레이아웃                     |                                      |
| AutoScaleMode            | Dpi                                  |
| AutoScroll               | True                                 |
| AutoScrollMargin         | 0, 0                                 |
| Size                     | 240, 266                             |
| WindowState              | Normal                               |
| 모양                       |                                      |
| BackColor                | Window                               |
| Font                     | Segoe Condensed, 10pt, style=Bold    |
| ForeColor                | ControlText                          |
| Text                     | Form1                                |
| 장 스타일                    |                                      |
| Icon                     | (아이콘)                                |
| Menu                     | mainMenu1                            |
| TopMost                  | False                                |



# 통지(Notifications (in a managed system))

- .NET 컴팩트 프레임워크에서 고급 P/Invoke (Platform Invoke) – <http://msdn.microsoft.com/en-us/library/aa446529.aspx>
- CeSetUserNotificationEx API에 대한 Wrapper를 만들
- P/Invoke 를 이용하여 그 wrapper (DLL 형태) 와 통신함

# 요약 및 심화 이슈들

- Reference:
  - C#.NET MOBILE PROGRAMMING – 안원국
  - Programming Windows CE.NET- Douglas Boling
- Windows CE 네트워킹
- 디바이스 간 통신
- 데스크탑과 연결
- 게임 API