dynamic 형식

이것이 C#이다

Contents

- dynamic 형식 소개
- 17.2 COM과 .NET의 상호 운용성을 위한 dynamic 형식
- 동적 언어와의 상호 운용성을 위한 dynamic 형식

17.1 dynamic 형식 소개

❖ 형식 검사가 프로그램 실행 중에 이루어지는 데이터 형식

- 컴파일러가 dynamic 키워드를 만나면 형식 검사를 실행때로 미룸

```
class MyClass
{
    public void FuncAAA()
    {  // Do Nothing   }
}

class MainApp
{
    static void Main(string[] args)
    {
        dynamic obj = new MyClass();
        obj.FuncAAA();
        obj.FuncBBB();
    }
}
```

dynamic 형식으로 선언된 obj는 일단 컴파일러의 형식 검사는 피해갑니다.

17.1.1 오리 타이핑

```
class Duck
{
    public void Walk()
    { Console.WriteLine("Duck.Walk"); }
    public void Swim()
    { Console.WriteLine("Duck.Swim"); }
    public void Quack()
    { Console.WriteLine("Duck.Quack"); }
}
class Robot
{
    public void Walk()
    { Console.WriteLine("Robot.Walk"); }
    public void Swim()
    { Console.WriteLine("Robot.Swim"); }
    public void Quack()
    { Console.WriteLine("Robot.Quack"); }
}
```

Duck도 오리이고,

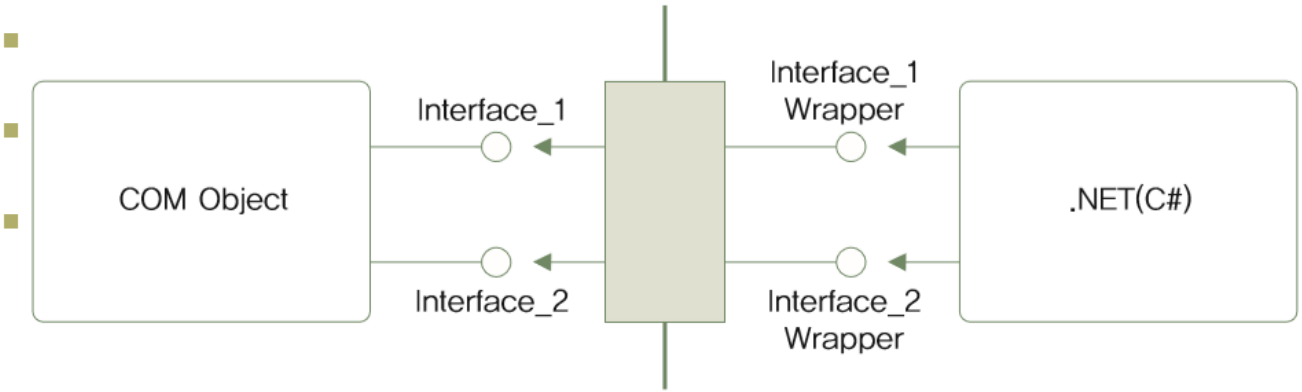
Robot도 오리입니다.

17.1.2 오리 타이핑의 장점과 단점

- **잘못된 인터페이스 설계로 인한 파생 클래스와 형제 클래스의 수정 문제**
  - 상속 관계를 이용하지 않아 프로그램의 동작 관련 부분만 수정 가능
- **Visual Studio의 리팩터링 기능 이용 불가**
  - 직접 모든 메소드 선언을 찾아서 수정

17.2 COM과 .NET의 상호 운용성을 위한 dynamic 형식(1)

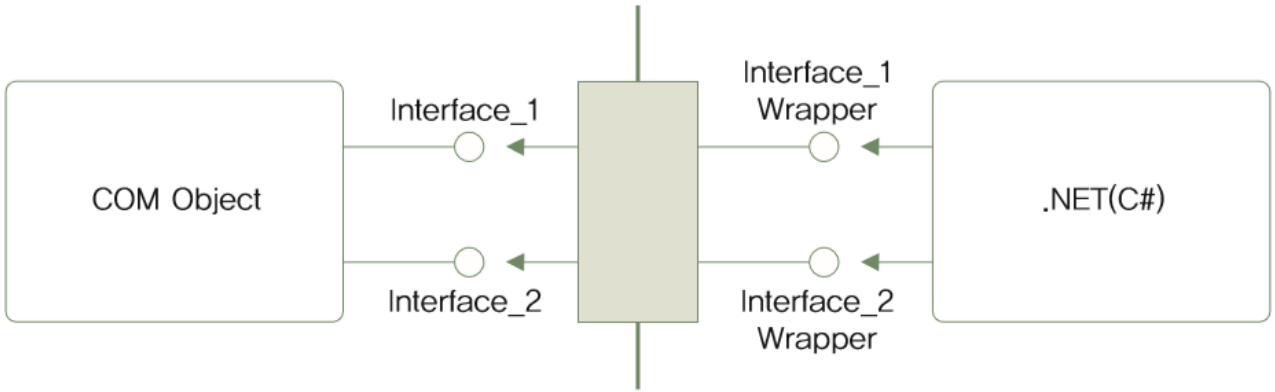
- **COM은 오래전에 등장한 부품 역할을 하는 소프트웨어**
  - 이미 만들어진 훌륭한 COM이 많다.
- **.NET 언어에서 RCW를 통해 COM 사용**
  - e)를 이용
  - 생성



- **C#과 COM 사이의 상호 운용성을 좋지 않게 만든 원인**
  - COM은 메소드가 결과를 반환할 때 실제 형식이 아닌 object 형식으로 반환
  - → C# 4.0에서 dynamic 도입해 해결
  - COM은 오버로딩을 지원하지않고 메소드의 선택적 매개 변수와 기본값 매개 변수 지원
  - →C#은 4.0에서 선택적 매개 변수와 기본값 매개 변수 지원으로 해결

17.2 COM과 .NET의 상호 운용성을 위한 dynamic 형식(1)

- **COM은 오래전에 등장한 부품 역할을 하는 소프트웨어**
  - 이미 만들어진 훌륭한 COM이 많다.
- **.NET 언어에서 RCW를 통해 COM 사용**
  - e)를 이용
  - 생성



- **C#과 COM 사이의 상호 운용성을 좋지 않게 만든 원인**
  - COM은 메소드가 결과를 반환할 때 실제 형식이 아닌 object 형식으로 반환
  - → C# 4.0에서 dynamic 도입해 해결
  - COM은 오버로딩을 지원하지않고 메소드의 선택적 매개 변수와 기본값 매개 변수 지원
  - →C#은 4.0에서 선택적 매개 변수와 기본값 매개 변수 지원으로 해결

17.2 COM과 .NET의 상호 운용성을 위한 dynamic 형식(2)

❖ C# 3.0과 4.0의 COM 상호 운용성 비교

C# 3.0 이하

```
public static void OldWay(string[,] data, string savePath)
{
    Excel.Application excelApp = new Excel.Application();

    excelApp.Workbooks.Add(Type.Missing);

    Excel.Worksheet workSheet =
        (Excel.Worksheet)excelApp.ActiveSheet;

    for (int i = 0; i < data.GetLength(0); i++)
    {
        ((Excel.Range)workSheet.Cells[i + 1, 1]).Value2 =
                                        data[i, 0];
        ((Excel.Range)workSheet.Cells[i + 1, 2]).Value2 =
                                        data[i, 1];
    }

    workSheet.SaveAs(savePath +
        "\\shpark-book-old.xlsx",
        Type.Missing,
        Type.Missing,
        Type.Missing,
        Type.Missing,
        Type.Missing,
        Type.Missing,
        Type.Missing,
        Type.Missing);

    excelApp.Quit();
}
```

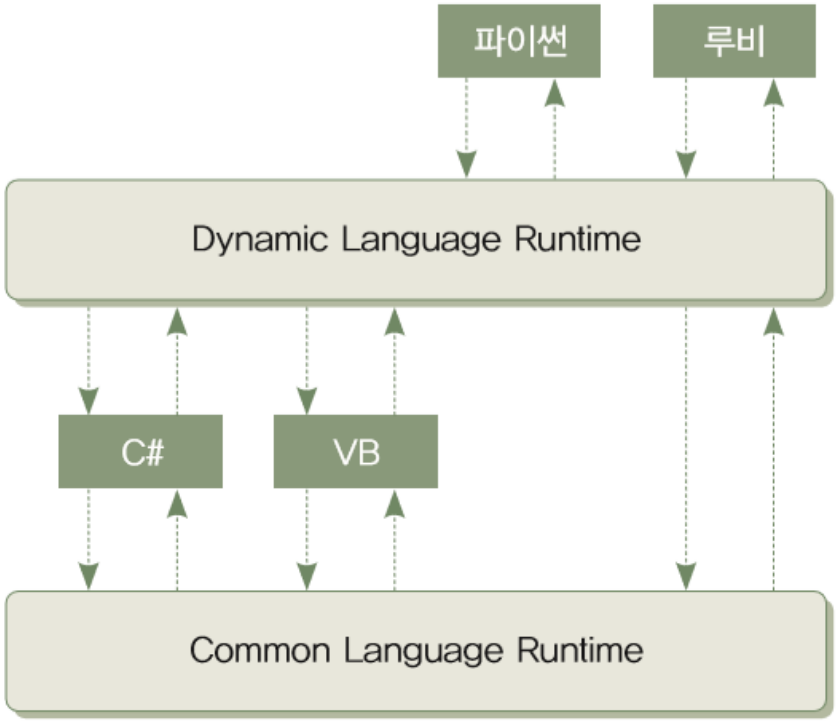
C# 4.0 이상

```
public static void NewWay(string[,] data, string savePath)
{
    Excel.Application excelApp = new Excel.Application();

    excelApp.Workbooks.Add();

    Excel._Worksheet workSheet = excelApp.ActiveSheet;

    for (int i = 0; i < data.GetLength(0); i++)
    {
        workSheet.Cells[i + 1, 1] = data[i, 0];
        workSheet.Cells[i + 1, 2] = data[i, 1];
    }

    workSheet.SaveAs(savePath +
           "\\shpark-book-dynamic.xlsx");
    excelApp.Quit();
}
```

❖ 데모 예제 - COMInterop

17.3 동적 언어와의 상호 운용성을 위한 dynamic 형식(1)

- **파이썬/루비처럼 실행 시 코드를 해석해 실행하는 동적 언어 지원**
  - 동적 언어 실행 플랫폼 DLR(CLR 위에서 동작)
  - 동적 언어에서 만든 객체에 C#/VB 같은 정적 언어의 접근 가능
  - 미리 형식 검사를 할 수 없는 동적 형식 언어에서 만든 객체를 C#의 dynamic 형식으로 해결



17.3 동적 언어와의 상호 운용성을 위한 dynamic 형식(2)

❖ C# 코드에서 동적 언어를 호스팅하는데 필요한 클래스

- ScriptRuntime
- ScriptScope
- ScriptEngine
- ScriptSource
- CompiledCode

❖ 게스트 코드를 실행하는 방법

- 소스 코드 "파일"의 경로를 넘겨받아 실행 - ScriptRuntime

```
ScriptRuntime runtime = Python.CreateRuntime();
dynamic result = runtime.ExecuteFile("namecard.py");
```

py는 파이썬 소스 코드의 확장자입니다.

- 문자열에 담긴 동적 언어 코드 실행 - ScriptEngine, ScriptScope, ScriptSource

❖ 데모 예제 – WithPython

Thank You !

이것이 C#이다