대리자와 이벤트

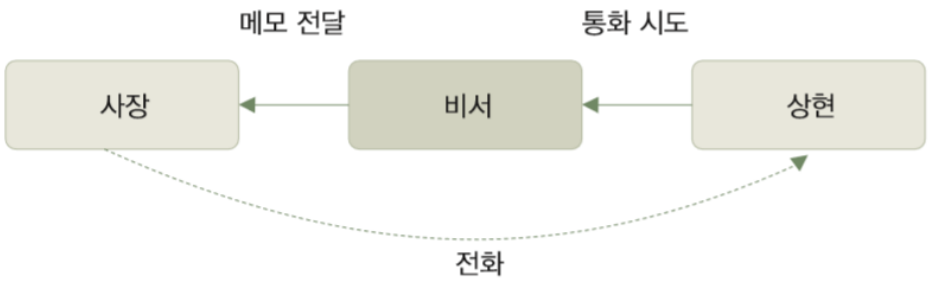
이것이 C#이다

Contents

- 대리자란?
- 대리자는 왜, 그리고 언제 사용하나요?
- 일반화 대리자
- 대리자 체인
- 익명 메소드
- 이벤트: 객체에 일어난 사건 알리기
- 대리자와 이벤트

13.1 대리자란? (1)

❖ 콜백



- 대신 어떤 일을 해줄 코드를 두고, 세부 실행 코드는 컴파일 시점이 아닌 실행 시점에 부여

❖ 콜백을 구현하는 방법 → 대리자

❖ 대리자는 메소드에 대한 참조
- 대리자에 메소드의 주소를 할당
- 대리자 호출 → 대리자 → 메소드 호출

❖ 선언 형식
- 한정자 delegate 반환 형식 대리자이름 (매개 변수_목록);

❖ 대리자는 인스턴스가 아닌 형식 → 인스턴스 만든 후 메서드 참조

13.1 대리자란? (2)

❖ 대리자를 이용한 콜백 구현 과정

- 대리자 선언
- 대리자의 인스턴스 생성
- 대리자 호출

```
delegate int MyDelegate( int a, int b );
```

```
int Plus( int a, int b )
{
    return a + b;
}
```

```
MyDelegate Callback;

Callback = new MyDelegate( Plus );
Console.WriteLine( Callback( 3, 4 ) );  // 7 출력

Callback = new MyDelegate( Minus );
Console.WriteLine( Callback( 7, 5 ) ); // 2 출력
```

대리자의 인스턴스를 만들 때도 new 연산자가 필요합니다.

❖ 데모 예제 - Delegate

13.2 대리자는 왜, 그리고 언제 사용하나요?

- “값” 이 아닌 “코드” 자체를 매개 변수로 넘기고 싶을 때
- 대리자를 사용 사례
  - 대리자 선언
  - 대리자가 참조할 비교 메소드 작성
  - 정렬할 배열과 대리자(비교 메소드 참조)를 매개변수로 받는 정렬 메소드 작성
  - 정렬 메소드 호출

```
delegate int Compare(int a, int b);

static int AscendCompare(int a, int b)

static void BubbleSort(int[] DataSet, Compare Comparer)
{
    int i = 0;
    int j = 0;
    int temp = 0;
    for ( i=0; i<DataSet.Length-1; i++ )
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if ( Comparer( DataSet[j] , DataSet[j+1] ) > 0 )
            {
                temp = DataSet[j+1];
                DataSet[j+1] = DataSet[j];
                DataSet[j] = temp;
            }
```

Comparer가 어떻게 구현된 메소드를 참조하고 있는가에 따라 정렬 결과가 달라집니다.

```
int[] array = { 3, 7, 4, 2, 10 };
BubbleSort(array, new Compare(AscendComparer)); // array는 { 2, 3, 4, 7, 10 }
```

- 데모 예제 - UsingCallback

13.3 일반화 대리자

- 일반화 메소드를 참조하는 대리자
- 일반화 대리자 사용 사례
  - 일반화 대리자 선언
  - 일반화 대리자가 참조할 일반화 메소드 구현
  - 정렬할 배열과 일반화 대리자를 매개변수로 받는 일반화 메소드 작성
- 데모 예제 – GenericDelegate

```
delegate int Compare<T>(T a, T b);
```

```
static int AscendCompare<T>(T a, T b) where T : IComparable<T>
{
    return a.CompareTo(b);
}
```

```
static void BubbleSort<T>( T[] DataSet, Compare<T> Comparer )
{
    int i = 0;
    int j = 0;
    T temp;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if (Comparer(DataSet[j], DataSet[j + 1]) > 0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

형식 매개 변수가 추가되었습니다.

13.4 대리자 체인

- 대리자 하나가 여러 개의 메소드를 동시에 참조
- 대리자 체인 연산자
  - 체인 연결: += 연산자
  - 체인 끊기: -= 연산자

```
delegate void ThereIsAFire( string location );
void Call119( string location )
{
    Console.WriteLine("소방서죠? 불났어요! 주소는 {0}", location);
}
```

```
ThereIsAFire Fire = new ThereIsAFire ( Call119 );
Fire += new ThereIsAFire ( ShotOut );
Fire += new ThereIsAFire ( Escape );
```

```
Fire( "우리집" );
```

Fire를 호출하면 다음을 출력합니다.

소방서죠? 불났어요! 주소는 우리집
피하세요! 우리집에 불이 났어요!
우리집에서 나갑시다!

- 여러 개의 콜백을 동시에 호출해야 할 때 유용
- 데모 예제 – DelegateChains

13.5 익명 메소드

- delegate 키워드를 이용하여 선언
- 익명 메소드 형식 = 참조할 대리자의 형식
- 선언 형식

```
대리자 인스턴스 = delegate ( 매개변수_목록 )
                  {
                  // 실행하고자 하는 코드 ...
                  }
```

- 사용 사례

```
delegate int Calculate( int a, int b );
```

```
Calculate Calc;

Calc = delegate ( int a, int b)
{
    return a + b;
}

Console.WriteLine( "3 + 4 : {0}", Calc( 3, 4 ) );
```

이름을 제외한 메소드의 구현.
이것이 익명 메소드입니다.

Calc를 호출하면 이 코드를 실행합니다.

- 데모 예제 - AnonymousMethod

13.6 이벤트: 객체에 일어난 사건 알리기

- 어떤 일이 생겼을 때 이를 알려주는 객체를 만들 때 사용
- 이벤트 선언과 사용 절차
  1. 대리자 선언
  2. 선언한 대리자 인스턴스를 event 한정자로 수식
  3. 이벤트 핸들러 작성
  4. 클래스 인스턴스 생성 후 객체의 이벤트에 이벤트 핸들러 등록
  5. 이벤트 발생 → 이벤트 핸들러 호출
- 사용 사례

```
class MyNotifier
{
static void Main(string[] args)
{
    MyNotifier notifier = new MyNotifier();
    notifier.SomethingHappened += new EventHandler( MyHandler );

    for (int i = 1; i < 30; i++)
    {
        // ...
}
```

SomethingHappened 이벤트에 MyHandler() 메소드를 이벤트 핸들러로 등록합니다.

- 데모 예제 - EventTest

13.7 대리자와 이벤트

❖ 대리자와 이벤트의 차이점

- 이벤트는 클래스 외부에서 호출 불가 (public 이라도)
- 대리자는 public이나 internal인 경우 클래스 외부에서 호출 가능

```
delegate void EventHandler(string message);

class MyNotifier
{
    public event EventHandler SomethingHappened;
}

class MainApp
{
    static void Main(string[] args)
    {
        MyNotifier notifier = new MyNotifier();
        notifier.SomethingHappened ( "테스트" );
    }
}
```

에러! 이벤트는 객체 외부에서 직접 호출할 수 없습니다.

❖ 대리자와 이벤트의 용도

- 대리자 – 콜백
- 이벤트 – 객체의 상태 변화나 사건의 발생 통지

Thank You !

이것이 C#이다