

출처: <https://ikorin2.hatenablog.jp/entry/2020/08/15/144224>

방법

```
// using System.Runtime.CompilerServices;
RuntimeHelpers.RunClassConstructor( typeof (Hoge).TypeHandle);
```

아래는 설명

static 생성자가 호출되지 않는 상황

C# 구조체나 클래스는 인스턴스의 생성자와는 별도로 **static** 생성자를 쓸 수 있다.

```
public class Hoge
{
    // 인스턴스 생성자
    public Hoge() {...}

    // static 생성자
    static Hoge() {...}
}
```

일반적으로 **static** 생성자는 처음 인스턴스 생성자가 호출될 때 처음 한 번만 실행된다. 즉, 타입에 대해 정적 초기화 처리를 쓸 수 있다.

또한 언어 사양으로서 반드시 한 번만 불리는 것이 보장 되기 때문에 멀티스레드에서도 배타 제어 처리 등을 할 필요가 없고, 스레드 세이프하게 실행된다.

클래스의 경우 첫 번째 인스턴스가 **new** 되었을 때 반드시 호출 되지만, 구조체의 경우 인스턴스가 있음에도 불구하고 **static** 생성자가 실행되지 않을 수 있다.

```
public struct MyData
{
    public int Value;

    // static 생성자
    static MyData () => Console.WriteLine( "static ctor called !!" );
}
```

위와 같은 구조인 경우 아래 코드는 **static** 생성자가 호출되지 않는다.

```
// Case 1
var array = new MyData[ 10 ];
Console.WriteLine(array[ 3 ].Value);    // Value : 0

// Case 2
var data = new MyData();
Console.WriteLine(data.Value);    // Value : 0
```

Case1과 **Case2** 모두 컴파일 후 실행할 수 있으며 **MyData** 타입의 인스턴스가 존재하고 있음에도 불구하고 **static** 생성자는 실행되지 않는다. 구조체이므로 두 경우 모두 메모리 제로 초기화가 이루어지고 있을 뿐이다.

덧붙여서 클래스의 경우 **Case1**은 **null**로, 인스턴스가 존재하지 않기 때문에 **static** 생성자가 불리지 않는 것은 당연하고, **Case2**는 제대로 호출된다.

구조체의 정적 초기화 처리를 **static** 생성자에서 하는 경우, 인스턴스를 사용하기 전에 호출되기 원하는 경우, 수동으로 호출해야 한다.

수동 호출

아래와 같이 수동으로 호출 할 수 있다.

```
// using System.Runtime.CompilerServices;
RuntimeHelpers.RunClassConstructor(typeof(MyData).TypeHandle);
```

메소드 이름이 **RunClassConstructor** 라고 쓰여져 있지만, 구조체에도 사용할 수 있다.

이 수동으로 부르는 방법은 자동으로 호출되는 조건처럼 단 한 번만 스레드 안전하게 실행된다. 여러 번 호출해도 2번째 이후는 무시된다. 또한 이미 자동으로 불린 상태에서 실행해도 아무것도 실행되지 않는다.