Static Constructors (C# Programming Guide)

- 07/20/2015
- 2 minutes to read
- Contributors

A static constructor is used to initialize any static data, or to perform a particular action that needs to be performed once only. It is called automatically before the first instance is created or any static members are referenced.

C#

```
class SimpleClass
{

    static readonly long baseline;



    static SimpleClass()
    {
        baseline = DateTime.Now.Ticks;
    }
}
```

Static constructors have the following properties:

- A static constructor does not take access modifiers or have parameters.
- A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
- A static constructor cannot be called directly.
- The user has no control on when the static constructor is executed in the program.
- A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.
- Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the `LoadLibrary` method.
- If a static constructor throws an exception, the runtime will not invoke it a second time, and the type will remain uninitialized for the lifetime of the application domain in which your program is running.

  1

Example

In this example, class `Bus` has a static constructor. When the first instance of `Bus` is created (`bus1`), the static constructor is invoked to initialize the class. The sample output verifies that the static constructor runs only one time, even though two instances of `Bus` are created, and that it runs before the instance constructor runs.

C#

```
public class Bus
{


    protected static readonly DateTime globalStartTime;


    protected int RouteNumber { get; set; }



    static Bus()
    {
        globalStartTime = DateTime.Now;



        Console.WriteLine("Static constructor sets global start time to {0}",
            globalStartTime.ToLongTimeString());
    }


    public Bus(int routeNum)
    {
        RouteNumber = routeNum;
        Console.WriteLine("Bus #{0} is created.", RouteNumber);
    }


    public void Drive()
    {
        TimeSpan elapsedTime = DateTime.Now - globalStartTime;



        Console.WriteLine("{0} is starting its route {1:N2} minutes after global
start time {2}.",
                                this.RouteNumber,
                                elapsedTime.TotalMilliseconds,
                                globalStartTime.ToShortTimeString());
    }
}

class TestBus
{
    static void Main()
    {
```

```
        Bus bus1 = new Bus(71);

        Bus bus2 = new Bus(72);

        bus1.Drive();

        System.Threading.Thread.Sleep(25);

        bus2.Drive();

        System.Console.WriteLine("Press any key to exit.");
        System.Console.ReadKey();
    }
}
```

See Also

C# Programming Guide
Classes and Structs
Constructors
Static Classes and Static Class Members
Finalizers

Note

The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, we invite you to read our blog post.