Sql server, .net and c# video tutorial

Free C#, .Net and Sql server video tutorial for beginners and intermediate programmers.

Why is singleton class sealed

Suggested Videos
Part 1 - Introduction to Design Patters - Text - Slides
Part 2 - Singleton Design Pattern - Text - Slides

In this tutorial we will discuss

- A quick Recap of Singleton version which we have discussed in the previous session
- We will focus on why we used Sealed keyword
- Why we need to seal the singleton class

In our previous video we discussed

- Why we need to create a singleton class and how we can apply singleton design pattern to a class
- We have changed the class to restrict external instantiation by changing the public constructor to private and then we provided a public property to access this class
- We have proved that adding Private constructor will prevent us instantiating a new class
- We have further sealed down this class to avoid any inheritance

You might be wondering why we need to seal the class when a private constructor is present.

Let's first remove the sealed keyword and check that. Let's create another class called DerivedSingleton and Inherit the singleton class. Let's compile the code and look at that it has thrown an error saying Singleton is inaccessible due to its protection level. This error is because of private constructor.

Now you might be thinking that when a private constructor is restricting the inheritance then why we need to apply sealed keyword to the class.

Let's just move this new class inside the Singleton class. By moving this class inside the Singleton class it has now become nested or child class of the main singleton class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/// <summary>
/// First Singleton version
/// </summary>
namespace SingletonDemo
{
```


Complete Tutorials

How to become a full stack web developer

Cloud computing complete tutorial

Healthy food for healthy mind and body

JavaScript tutorial

Bootstrap tutorial

Angular tutorial for beginners

Angular 5 Tutorial for beginners

Important Videos

The Gift of Education

Web application for your business

How to become .NET developer

Resources available to help you

Dot Net Video Tutorials

Blazor tutorial

C tutorial

ASP.NET Core Tutorial

ASP.NET Core Razor Pages Tutorial

Angular 6 Tutorial

Angular CRUD Tutorial

Angular CLI Tutorial

Angular 2 Tutorial

Design Patterns

SOLID Principles

ASP.NET Web API

```
    /*
     *  Sealed restricts the inheritance
     */
    public class Singleton
    {
        private static int counter = 0;
        private static object obj = new object();
        /*
        * Private constructor ensures that object is not
        * instantiated other than with in the class itself
        */
        private Singleton()
        {
            counter++;
            Console.WriteLine("Counter Value " + counter.ToString());
        }
        private static Singleton instance = null;
        /*
         * public property is used to return only one instance of the class
         * leveraging on the private property
         */
        public static Singleton GetInstance
        {
            get
            {
                if (instance == null)
                    instance = new Singleton();
                return instance;
            }
        }
        /*
         * Public method which can be invoked through the singleton instance
         */
        public void PrintDetails(string message)
        {
            Console.WriteLine(message);
        }
        /*
         * By removing sealed keyword we can inherit the singleton and instantiate multiple
objects
         * This violates singleton design principles.
         */
        public class DerivedSingleton : Singleton
        {
        }
    }
}
```

What is a nested class?
A class with in another class is called a nested class.

Now that we have moved the derived class to nested class lets compile the program and check. Look at that we are able to compile this successfully.

Now, let's switch to main program and access the nested class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/// <summary>
/// First version of Singleton demo
/// </summary>
namespace SingletonDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            /*
            * Assuming Singleton is created from student class
            * we refer to the GetInstance property from the Singleton class
            */
            Singleton fromStudent = Singleton.GetInstance;
            fromStudent.PrintDetails("From Student");
            /*
            * Assuming Singleton is created from employee class
            * we refer to the GetInstance property from the Singleton class
```

Bootstrap
AngularJS Tutorial
jQuery Tutorial
JavaScript with ASP.NET Tutorial
JavaScript Tutorial
Charts Tutorial
LINQ
LINQ to SQL
LINQ to XML
Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#
SQL Server
ADO.NET
ASP.NET
GridView
ASP.NET MVC
Visual Studio Tips and Tricks
Dot Net Interview Questions

Slides
Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#
SQL Server
ADO.NET
ASP.NET
GridView
ASP.NET MVC
Visual Studio Tips and Tricks

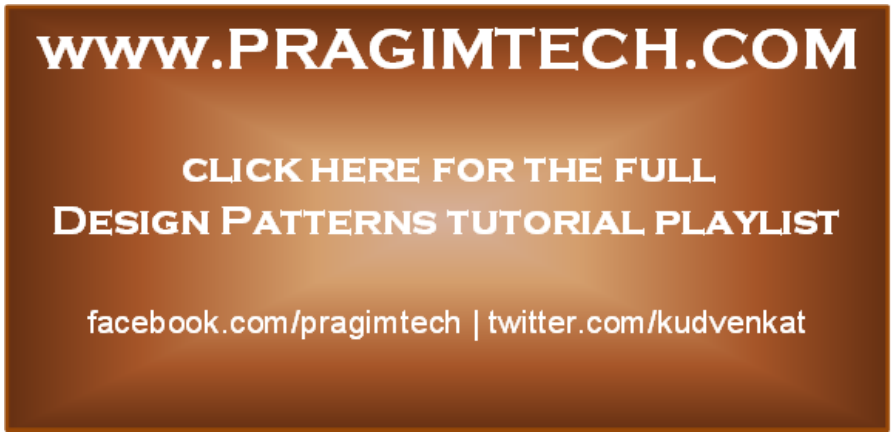
```
            */
            Singleton fromEmployee = Singleton.GetInstance;
            fromEmployee.PrintDetails("From Employee");

            Console.WriteLine("-------------------------------------");
            /*
             * Instantiating singleton from a derived class. This violates singleton pattern
principles.
             */
            Singleton.DerivedSingleton derivedObj = new Singleton.DerivedSingleton();
            derivedObj.PrintDetails("From Derived");

            Console.ReadLine();
        }
    }
}
```

Lets run the program. Look at that the counter value has incremented to 2 proving that we are able to create multiple instances of the singleton using the nested derived class.

This violates the principle of singleton. Let's go back to the Singleton and make the class as sealed. Let's compile the program

Look at that we have got an error when we compile the program saying we cannot derive a sealed class. With this we have proved that private constructor helps in preventing any external instantiations of objects and sealed will prevent the class inheritances.

In the next session we will discuss how to handle thread safety in singleton as the current version can create multiple instances in multi-threaded environments.



3 comments:

Sunil Bailwal November 28, 2017 at 5:17 PM

Hi,

In the above program we are using the below code.

private static object obj = new object();

but we are not using obj anywhere in this scenario. We are using the above obj object in the topic 'Thread safety in Singleton' to lock the object in multiple thread environment.

Reply

GSB January 21, 2018 at 9:13 AM

Yes. He must have written the code at once and then forgot to remove this object in this part. Nothing to worry about I guess.

Reply

Java Video Tutorials

Part 1 : Video | Text | Slides

Part 2 : Video | Text | Slides

Part 3 : Video | Text | Slides

Interview Questions

C#

SQL Server

Written Test

Add Surf Earn December 24, 2018 at 11:44 PM

i have modified the below code with sealed keyword scenario
public class DerivedSingleton
{
void f1() {
SingleTon obj = new SingleTon();
}
}
from outside of singleton class now we are able to call the method singleton.DerivedSingleton.f1() [by creating derviedsingleton class object] so now instance count will goes more than one then singleton principle will break right ?? please correct me if i am wrong

Reply

Enter Comment

It would be great if you can help share these free resources

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)