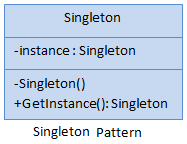
Singleton pattern falls under Creational Pattern of [Gang of Four (GOF) Design Patterns in .Net](http://www.dotnettricks.com/learn/designpatterns/gang-of-four-gof-design-patterns-in-net). It is pattern is one of the simplest design patterns. This pattern ensures that a class has only one instance.

What is Singleton Pattern?

Singleton pattern is one of the simplest design patterns. This pattern ensures that a class has only one instance and provides a global point of access to it.

Singleton Pattern - UML Diagram & Implementation

The UML class diagram for the implementation of the Singleton design pattern is given below:



The classes, and objects in the above UML class diagram are as follows:

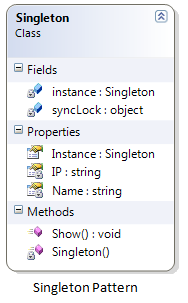
1. Singleton

This is a class which is responsible for creating and maintaining its own unique instance.

C# - Implementation Code

1. *//eager initialization of singleton*
2. public class Singleton
3. {
4. private static Singleton instance = new Singleton();
5. private Singleton() { }
7. public static Singleton GetInstance
8. {
9. get
10. {
11. return instance;
12. }
13. }
14. }
16. *////lazy initialization of singleton*
17. public class Singleton
18. {
19. private static Singleton instance = null;
20. private Singleton() { }
22. public static Singleton GetInstance
23. {
24. get
25. {
26. if (instance == null)
27. instance = new Singleton();
29. return instance;
30. }
31. }
32. }
34. *////Thread-safe (Double-checked Locking) initialization of singleton*
35. public class Singleton
36. {
37. private static Singleton instance = null;
38. private Singleton() { }
39. private static object lockThis = new object();
41. public static Singleton GetInstance
42. {
43. get
44. {
45. lock (lockThis)
46. {
47. if (instance == null)
48. instance = new Singleton();
50. return instance;
51. }
52. }
53. }
54. }

Singleton Pattern - Example



Who is what?

The classes and objects in the above class diagram can be identified as follows:

1. **Singleton** - Singleton class

C# - Sample Code

1. */// <summary>*
2. */// The 'Singleton' class*
3. */// </summary>*
4. public class Singleton
5. {
6. *// .NET guarantees thread safety for static initialization*
7. private static Singleton instance = null;
8. private string Name{get;set;}
9. private string IP{get;set;}
10. private Singleton()
11. {
12. *//To DO: Remove below line*
13. Console.WriteLine("Singleton Intance");
15. Name = "Server1";
16. IP = "192.168.1.23";
17. }
18. *// Lock synchronization object*
19. private static object syncLock = new object();
21. public static Singleton Instance
22. {
23. get
24. {
25. *// Support multithreaded applications through*
26. *// 'Double checked locking' pattern which (once*
27. *// the instance exists) avoids locking each*
28. *// time the method is invoked*
29. lock (syncLock)
30. {
31. if (Singleton.instance == null)
32. Singleton.instance = new Singleton();
34. return Singleton.instance;
35. }
36. }
37. }
39. public void Show()
40. {
41. Console.WriteLine("Server Information is : Name={0} & IP={1}", IP, Name);
42. }
44. }
46. */// <summary>*
47. */// Singleton Pattern Demo*
48. */// </summary>*
49. *///*
50. class Program
51. {
52. static void Main(string[] args)
53. {
54. Singleton.Instance.Show();
55. Singleton.Instance.Show();
57. Console.ReadKey();
58. }
59. }

Singleton Pattern Demo - Output

https://dotnettricks.blob.core.windows.net/img/designpatterns/singleton2.png

When to use it?

1. Exactly one instance of a class is required.
2. Controlled access to a single object is necessary.