عنوان: **نحوه ایجاد الگوی Singleton به صورت جنریک**
نویسنده: احمد نواصری
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۳:۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Design patterns

در برخی از مواقع، ایجاد یک وهله از یک کلاس کاری هزینه بر می‌باشد. بنابراین نیاز است تا فقط یک وهله از آن کلاس را ایجاد و تا آخر اجرای برنامه از آن استفاده کرد. این راه حل در قالب یک الگوی طراحی به نام Singleton معرفی شده است. حال می‌خواهیم با استفاده از امکانات جنریک، کلاسی را طراحی کنیم تا عملیات ساخت وهله‌ها را انجام دهد.
نکاتی که در طراحی یک الگوی Singleton باید مد نظر داشت این است که:
دسترسی سازنده کلاس Singleton را از نوع Private تعیین کنیم.
یک فیلد استاتیک از نوع کلاس Singleton تعریف کنیم.
یک خاصیت از نوع استاتیک فقط خواندنی (یعنی فقط get داشته باشد) تعریف کرده تا فیلد استاتیک را مقداردهی و Return کند. به جای پروپرتی میتوان از یک متد استاتیک نیز استفاده کرد.

```
public class SingletonClassCreator<T> where T:class , new()
    {
        private static T _singletoneInstance;
        private static readonly object Lock = new object();

        public static T SingletoneInstance
        {
            get
            {
                lock (Lock)
                {
                    if (_singletoneInstance == null)
                    {
                        _singletoneInstance = new T();
                    }
                }
                return _singletoneInstance;
            }
        }

        private SingletonClassCreator()
        {
        }
    }
```

برای ایجاد حالت Tread-Safe در برنامه هایی که امکان دسترسی همزمان به یک شیء (مثلا در برنامه‌های وب) وجود دارد، از یک بلاک Lock استفاده شده است تا در هر لحظه فقی یک نخ قادر به ایجاد Singleton شود.
حال برای ایجاد وهله‌های Singleton از کلاسهای مورد نظر به صورت زیر عمل میکنیم

```
public class FirstSingleton
    {
        public int Square(int input)
        {
            return input*input;
        }
    }
```

```
static void Main(string[] args)
        {
            var firstSingletone = SingletonClassCreator<FirstSingleton>.SingletoneInstance ;
            Console.WriteLine(firstSingletone.Square(12));
            Console.ReadKey();
        }
```

در خط اول، با تعریف یک متغیر و قرار دادن وهله استاتیک که بوسیله پروپرتی استاتیک SingletoneInstance برگشت داده میشود، یک شی Singleton از کلاس FirstSingleton را ایجاد میکنیم.

نظرات خوانندگان

نویسنده: مصطفی
تاریخ: ۱۳۹۳/۰۸/۲۷ ۹:۳۳

یه روش بهتر برای استفاده در حالت Thread Safe که به نظرم بهینه‌تر هستش در زمان اجرا, بهینه سازی کد به این شکل هستش

```
if (instance == null)
         {
            lock (Lock)
            {
               if (instance == null)
                  instance = new Singleton();
            }
         }
```

با این حالت در صورتی که شی قبلا ایجاد شده باشه هیچ کدوم از Threadها رو بلوک نمیکنه.

نویسنده: مصطفی
تاریخ: ۱۳۹۳/۰۸/۲۷ ۹:۳۴

همچنین میشد این کلاس رو با این لینک تلفیق کرد که یه خروجی جالبتری به وجود بیاد.
با تشکر از زحمت شما