

LinksPlatform's Platform.Singletons Class Library

./Default[T].cs

```
1 using System;
2
3 #pragma warning disable RECS0017 // Possible compare of value type with 'null'
4
5 namespace Platform.Singletons
6 {
7     /// <summary>
8     /// <para>Represents an access point to instances of default types (created using the
9     /// <para>↳ constructor with no arguments).</para>
10    /// <para>Представляет собой точку доступа к экземплярам типов по умолчанию (созданных с
11    /// <para>↳ помощью конструктора без аргументов).</para>
12    /// </summary>
13    /// <typeparam name="T"><para>The type of instance of the object.</para><para>Тип экземпляра
14    /// <para>↳ объекта.</para></typeparam>
15    public static class Default<T>
16    where T : new()
17    {
18        [ThreadStatic]
19        private static T _threadInstance;
20
21        /// <summary>
22        /// <para>Returns an instance of an object by default.</para>
23        /// <para>Возвращает экземпляр объекта по умолчанию.</para>
24        /// </summary>
25        public static readonly T Instance = new T();
26
27        /// <summary>
28        /// <para>If you really need maximum performance, use this property. This property
29        /// <para>↳ should create only one instance per thread.</para>
30        /// <para>Если вам действительно нужна максимальная производительность, используйте это
31        /// <para>↳ свойство. Это свойство должно создавать только один экземпляр на поток.</para>
32        /// </summary>
33        /// <remarks>
34        /// <para>Check for null is intended to create only classes, not structs.</para>
35        /// <para>Проверка на значение null выполняется специально для создания только классов,
36        /// <para>↳ а не структур.</para>
37        /// </remarks>
38        public static T ThreadInstance => _threadInstance == null ? _threadInstance = new T() :
39        _threadInstance; //-V3111
40    }
41 }
```

./Global.cs

```
1 namespace Platform.Singletons
2 {
3     /// <summary>
4     /// <para>Contains the global state of the system.</para>
5     /// <para>Содержит глобальное состояние системы.</para>
6     /// </summary>
7     public static class Global
8     {
9         /// <summary>
10        /// <para>
11        /// <para>↳ Represents a garbage field where you can dump unnecessary values.
12        /// <para>↳ In some cases, this may help to avoid unwanted optimization and pretend that the
13        /// <para>↳ value is really used.
14        /// <para>↳ This may be useful when implementing performance tests.
15        /// </para>
16        /// <para>Представляет поле-помойку, куда можно сбрасывать ненужные значения.
17        /// <para>↳ В некоторых случаях это может помочь избежать нежелательной оптимизации и сделать
18        /// <para>↳ вид, что значение действительно используется.
19        /// <para>↳ Такое может быть полезно при реализации тестов на производительность.
20        /// </para>
21        /// </summary>
22        public static object Trash { get; set; }
23    }
24 }
```

./Singleton.cs

```
1 using System;
2 using Platform.Interfaces;
3
4 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6 namespace Platform.Singletons
```

```

7 {
8     public static class Singleton
9     {
10         public static Singleton<T> Create<T>(Func<T> creator) => new Singleton<T>(creator);
11         public static Singleton<T> Create<T>(IFactory<T> factory) => new
            ↳ Singleton<T>(factory.Create);
12         public static T Get<T>(Func<T> creator) => new Singleton<T>(creator).Instance;
13         public static T Get<T>(IFactory<T> factory) => new Singleton<T>(factory.Create).Instance;
14     }
15 }

```

./Singleton[T].cs

```

1 using System;
2 using System.Collections.Concurrent;
3 using System.Reflection;
4 using Platform.Collections.Lists;
5 using Platform.Reflection;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8 #pragma warning disable RECS0108 // Warns about static fields in generic types
9
10 namespace Platform.Singletons
11 {
12     public struct Singleton<T>
13     {
14         private static readonly ConcurrentDictionary<Func<T>, byte[]> _functions = new
            ↳ ConcurrentDictionary<Func<T>, byte[]>();
15         private static readonly ConcurrentDictionary<byte[], T> _singletons = new
            ↳ ConcurrentDictionary<byte[], T>(Default<IListEqualityComparer<byte[]>>.Instance);
16         public T Instance { get; }
17         public Singleton(Func<T> creator) => Instance =
            ↳ _singletons.GetOrAdd(_functions.GetOrAdd(creator,
            ↳ creator.GetMethodInfo().GetILBytes()), key => creator());
18     }
19 }

```

Index

./Default[T].cs, 1
./Global.cs, 1
./Singleton.cs, 1
./Singleton[T].cs, 2