

LinksPlatform's Platform.Singletons Class Library

./Default[T].cs

```
1 using System;
2
3 namespace Platform.Singletons
4 {
5     /// <summary>
6     /// <para>Represents an access point to instances of default types (created using the
7     ///     ↪ constructor with no arguments).</para>
8     /// <para>Представляет собой точку доступа к экземплярам типов по умолчанию (созданных с
9     ///     ↪ помощью конструктора без аргументов).</para>
10    /// </summary>
11    /// <typeparam name="T"><para>The type of instance of the object.</para><para>Тип экземпляра
12    ///     ↪ объекта.</para></typeparam>
13    public static class Default<T>
14    where T : new()
15    {
16        [ThreadStatic]
17        private static T _threadInstance;
18
19        /// <summary>
20        /// <para>Returns an instance of an object by default.</para>
21        /// <para>Возвращает экземпляр объекта по умолчанию.</para>
22        /// </summary>
23        public static readonly T Instance = new T();
24
25        /// <summary>
26        /// <para>If you really need maximum performance, use this property. This property
27        ///     ↪ should create only one instance per thread.</para>
28        /// <para>Если вам действительно нужна максимальная производительность, используйте это
29        ///     ↪ свойство. Это свойство должно создавать только один экземпляр на поток.</para>
30        /// </summary>
31        public static T ThreadInstance => _threadInstance == null ? _threadInstance = new T() :
32            ↪ _threadInstance; //-V3111
33    }
34 }
```

./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        ///     Represents a garbage field where you can dump unnecessary values.
12        ///     In some cases, this may help to avoid unwanted optimization and pretend that the
13        ///     ↪ value is really used.
14        ///     This may be useful when implementing performance tests.
15        /// </para>
16        /// <para>
17        ///     Представляет поле-помойку, куда можно сбрасывать ненужные значения.
18        ///     В некоторых случаях это может помочь избежать нежелательной оптимизации и сделать
19        ///     ↪ вид, что значение действительно используется.
20        ///     Такое может быть полезно при реализации тестов на производительность.
21        /// </para>
22        /// </summary>
23        public static object Trash { get; set; }
24    }
25 }
```

./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
12            ↪ Singleton<T>(factory.Create);
13        public static T Get<T>(Func<T> creator) => new Singleton<T>(creator).Instance;
14    }
15 }
```

```

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
9  namespace Platform.Singletons
10 {
11     public struct Singleton<T>
12     {
13         private static readonly ConcurrentDictionary<Func<T>, byte[]> _functions = new
14             ↳ ConcurrentDictionary<Func<T>, byte[]>();
15         private static readonly ConcurrentDictionary<byte[], T> _singletons = new
16             ↳ ConcurrentDictionary<byte[], T>(Default<IListEqualityComparer<byte[]>>.Instance);
17         public T Instance { get; }
18         public Singleton(Func<T> creator) => Instance =
19             ↳ _singletons.GetOrAdd(_functions.GetOrAdd(creator,
20             ↳ creator.GetMethodInfo().GetILBytes()), key => creator());
21     }
22 }

```

Index

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