# LinksPlatform's Platform.Singletons Class Library

## ./Default[T].cs

```
using System;

namespace Platform.Singletons
{
    /// <summary>
    /// <para>Represents an access point to instances of default types (created using the constructor with no arguments).</para>
    /// <para>Представляет собой точку доступа к экземплярам типов по умолчанию (созданных с помощью конструктора без аргументов).</para>
    /// </summary>
    /// <typeparam name="T"><para>The type of instance of the object.</para><para>Тип экземпляра объекта.</para></typeparam>
    public static class Default<T>
        where T : new()
    {
        [ThreadStatic]
        private static T _threadInstance;

        /// <summary>
        /// <para>Returns an instance of an object by default.</para>
        /// <para>Возвращает экземпляр объекта по умолчанию.</para>
        /// </summary>
        public static readonly T Instance = new T();

        /// <summary>
        /// <para>If you really need maximum performance, use this property. This property should create only one instance per thread.</para>
        /// <para>Если вам действительно нужна максимальная производительность, используйте это свойство. Это свойство должно создавать только один экземпляр на поток.</para>
        /// </summary>
        public static T ThreadInstance => _threadInstance == null ? _threadInstance = new T() : _threadInstance; //-V3111
    }
}
```

## ./Global.cs

```
namespace Platform.Singletons
{
    /// <summary>
    /// <para>Contains the global state of the system.</para>
    /// <para>Содержит глобальное состояние системы.</para>
    /// </summary>
    public static class Global
    {
        /// <summary>
        /// <para>
        /// Represents a garbage field where you can dump unnecessary values.
        /// In some cases, this may help to avoid unwanted optimization and pretend that the value is really used.
        /// This may be useful when implementing performance tests.
        /// </para>
        /// <para>
        /// Представляет поле-помойку, куда можно сбрасывать ненужные значения.
        /// В некоторых случаях это может помочь избежать нежелательной оптимизации и сделать вид, что значение действительно используется.
        /// Такое может быть полезно при реализации тестов на производительность.
        /// </para>
        /// </summary>
        public static object Trash { get; set; }
    }
}
```

## ./Singleton.cs

```
using System;
using Platform.Interfaces;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Singletons
{
    public static class Singleton
    {
        public static Singleton<T> Create<T>(Func<T> creator) => new Singleton<T>(creator);
        public static Singleton<T> Create<T>(IFactory<T> factory) => new Singleton<T>(factory.Create);
        public static T Get<T>(Func<T> creator) => new Singleton<T>(creator).Instance;
```

```
        public static T Get<T>(IFactory<T> factory) => new Singleton<T>(factory.Create).Instance;
    }
}
```

./Singleton[T].cs

```
using System;
using System.Collections.Concurrent;
using System.Reflection;
using Platform.Collections.Lists;
using Platform.Reflection;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Singletons
{
    public struct Singleton<T>
    {
        private static readonly ConcurrentDictionary<Func<T>, byte[]> _functions = new
        ↪  ConcurrentDictionary<Func<T>, byte[]>();
        private static readonly ConcurrentDictionary<byte[], T> _singletons = new
        ↪  ConcurrentDictionary<byte[], T>(Default<IListEqualityComparer<byte>>.Instance);
        public T Instance { get; }
        public Singleton(Func<T> creator) => Instance =
        ↪  _singletons.GetOrAdd(_functions.GetOrAdd(creator,
        ↪  creator.GetMethodInfo().GetILBytes()), key => creator());
    }
}
```

## Index