

LinksPlatform's Platform.Singletons Class Library

1.1 ./csharp/Platform.Singletons/Default[T].cs

```
1 using System;
2 using System.Runtime.CompilerServices;
3
4 #pragma warning disable RECS0017 // Possible compare of value type with 'null'
5
6 namespace Platform.Singletons
7 {
8     /// <summary>
9     /// <para>Represents an access point to instances of default types (created using the
10     ///   ↳ constructor with no arguments).</para>
11     /// <para>Представляет собой точку доступа к экземплярам типов по умолчанию (созданных с
12     ///   ↳ помощью конструктора без аргументов).</para>
13     /// </summary>
14     /// <typeparam name="T"><para>The type of instance of the object.</para><para>Тип экземпляра
15     ///   ↳ объекта.</para></typeparam>
16     public static class Default<T>
17     where T : new()
18     {
19         [ThreadStatic]
20         private static T _threadInstance;
21
22         /// <summary>
23         /// <para>Returns an instance of an object by default.</para>
24         /// <para>Возвращает экземпляр объекта по умолчанию.</para>
25         /// </summary>
26         public static readonly T Instance = new T();
27
28         /// <summary>
29         /// <para>If you really need maximum performance, use this property. This property
30         ///   ↳ should create only one instance per thread.</para>
31         /// <para>Если вам действительно нужна максимальная производительность, используйте это
32         ///   ↳ свойство. Это свойство должно создавать только один экземпляр на поток.</para>
33         /// </summary>
34         /// <remarks>
35         /// <para>Check for null is intended to create only classes, not structs.</para>
36         /// <para>Проверка на значение null выполняется специально для создания только классов,
37         ///   ↳ а не структур.</para>
38         /// </remarks>
39         public static T ThreadInstance
40         {
41             [MethodImpl(MethodImplOptions.AggressiveInlining)]
42             get => _threadInstance == null ? _threadInstance = new T() : _threadInstance;
43             ↳ //-V3111
44         }
45     }
46 }
```

1.2 ./csharp/Platform.Singletons/Global.cs

```
1 using System.Runtime.CompilerServices;
2
3 namespace Platform.Singletons
4 {
5     /// <summary>
6     /// <para>Contains the global state of the system.</para>
7     /// <para>Содержит глобальное состояние системы.</para>
8     /// </summary>
9     public static class Global
10     {
11         /// <summary>
12         /// <para>
13         ///   ↳ Represents a garbage field where you can dump unnecessary values.
14         ///   ↳ In some cases, this may help to avoid unwanted optimization and pretend that the
15         ///   ↳ value is really used.
16         ///   ↳ This may be useful when implementing performance tests.
17         /// </para>
18         /// <para>
19         ///   ↳ Представляет поле-помойку, куда можно сбрасывать ненужные значения.
20         ///   ↳ В некоторых случаях это может помочь избежать нежелательной оптимизации и сделать
21         ///   ↳ вид, что значение действительно используется.
22         ///   ↳ Такое может быть полезно при реализации тестов на производительность.
23         /// </para>
24         /// </summary>
25         public static object Trash
26         {
27             [MethodImpl(MethodImplOptions.AggressiveInlining)]
28             get;
```

```

27         [MethodImpl(MethodImplOptions.AggressiveInlining)]
28         set;
29     }
30 }
31 }

```

1.3 ./csharp/Platform.Singletons/Singleton.cs

```

1 using System;
2 using System.Runtime.CompilerServices;
3 using Platform.Interfaces;
4
5 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7 namespace Platform.Singletons
8 {
9     public static class Singleton
10     {
11         [MethodImpl(MethodImplOptions.AggressiveInlining)]
12         public static Singleton<T> Create<T>(Func<T> creator) => new Singleton<T>(creator);
13
14         [MethodImpl(MethodImplOptions.AggressiveInlining)]
15         public static Singleton<T> Create<T>(IFactory<T> factory) => new
16             ↳ Singleton<T>(factory.Create);
17
18         [MethodImpl(MethodImplOptions.AggressiveInlining)]
19         public static T Get<T>(Func<T> creator) => Create(creator).Instance;
20
21         [MethodImpl(MethodImplOptions.AggressiveInlining)]
22         public static T Get<T>(IFactory<T> factory) => Create(factory).Instance;
23     }
24 }

```

1.4 ./csharp/Platform.Singletons/Singleton[T].cs

```

1 using System;
2 using System.Collections.Concurrent;
3 using System.Reflection;
4 using System.Runtime.CompilerServices;
5 using Platform.Collections.Lists;
6 using Platform.Reflection;
7
8 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
9 #pragma warning disable RECS0108 // Warns about static fields in generic types
10
11 namespace Platform.Singletons
12 {
13     public struct Singleton<T>
14     {
15         private static readonly ConcurrentDictionary<Func<T>, byte[]> _functions = new
16             ↳ ConcurrentDictionary<Func<T>, byte[]>();
17         private static readonly ConcurrentDictionary<byte[], T> _singletons = new
18             ↳ ConcurrentDictionary<byte[], T>(Default<IListEqualityComparer<byte[]>>.Instance);
19
20         public T Instance
21         {
22             [MethodImpl(MethodImplOptions.AggressiveInlining)]
23             get;
24         }
25
26         [MethodImpl(MethodImplOptions.AggressiveInlining)]
27         public Singleton(Func<T> creator) => Instance =
28             ↳ _singletons.GetOrAdd(_functions.GetOrAdd(creator,
29                 ↳ creator.GetMethodInfo().GetILBytes()), key => creator());
30     }
31 }

```

1.5 ./csharp/Platform.Singletons.Tests/DefaultTests.cs

```

1 using Xunit;
2
3 namespace Platform.Singletons.Tests
4 {
5     public class DefaultTests
6     {
7         [Fact]
8         public void StructInstanceTest()
9         {
10             Assert.Equal(0, Default<int>.Instance);
11         }
12
13         [Fact]

```

```

14     public void ClassInstanceTest()
15     {
16         Assert.NotNull(Default<object>.Instance);
17     }
18
19     [Fact]
20     public void StructThreadInstanceTest()
21     {
22         Assert.Equal(0, Default<int>.ThreadInstance);
23     }
24
25     [Fact]
26     public void ClassThreadInstanceTest()
27     {
28         Assert.NotNull(Default<object>.ThreadInstance);
29     }
30 }
31 }

```

1.6 ./csharp/Platform.Singletons.Tests/GlobalTests.cs

```

1  using Xunit;
2
3  namespace Platform.Singletons.Tests
4  {
5      public class GlobalTests
6      {
7          [Fact]
8          public void TrashIsNullTest()
9          {
10             Assert.Null(Global.Trash);
11         }
12     }
13 }

```

1.7 ./csharp/Platform.Singletons.Tests/SingletonTests.cs

```

1  using Xunit;
2
3  namespace Platform.Singletons.Tests
4  {
5      public class SingletonTests
6      {
7          [Fact]
8          public void TwoValuesAreTheSameTest()
9          {
10             var value1 = Singleton.Get(() => 1);
11             var value2 = Singleton.Get(() => 1);
12             Assert.Equal(value1, value2);
13         }
14
15         // Looks like ILBytes do not help here
16         //[Fact]
17         //public void TwoFunctionsAreTheSameTest()
18         //{
19             // //Func<Func<int>> factory = () => () => 1;
20             // var func1 = Singleton.Get<Func<int>>(() => () => 1);
21             // var func2 = Singleton.Get<Func<int>>(() => () => 1);
22             // Assert.Equal(func1, func2);
23         //}
24     }
25 }

```

Index

- ./csharp/Platform.Singletons.Tests/DefaultTests.cs, 2
- ./csharp/Platform.Singletons.Tests/GlobalTests.cs, 3
- ./csharp/Platform.Singletons.Tests/SingletonTests.cs, 3
- ./csharp/Platform.Singletons/Default[T].cs, 1
- ./csharp/Platform.Singletons/Global.cs, 1
- ./csharp/Platform.Singletons/Singleton.cs, 2
- ./csharp/Platform.Singletons/Singleton[T].cs, 2