

LinksPlatform's Platform.Interfaces Class Library

1.1 ./csharp/Platform.Interfaces/ICounter[TResult, TArgument].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a counter that requires an argument to perform a count.</para>
5     /// <para>Определяет счётчик, которому требуется аргумент для выполнения подсчёта.</para>
6     /// </summary>
7     /// <typeparam name="TArgument">
8     /// <para>The argument type.</para>
9     /// <para>Тип аргумента.</para>
10    /// </typeparam>
11    /// <typeparam name="TResult">
12    /// <para>The count result type.</para>
13    /// <para>Тип результата подсчёта.</para>
14    /// </typeparam>
15    public interface ICounter<out TResult, in TArgument>
16    {
17        /// <summary>
18        /// <para>Performs a count.</para>
19        /// <para>Выполняет подсчёт.</para>
20        /// </summary>
21        /// <param name="argument">
22        /// <para>The argument.</para>
23        /// <para>Аргумент.</para>
24        /// </param>
25        /// <returns>
26        /// <para>The count result.</para>
27        /// <para>Результат подсчёта.</para>
28        /// </returns>
29        TResult Count(TArgument argument);
30    }
31 }
```

1.2 ./csharp/Platform.Interfaces/ICounter[TResult].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a counter.</para>
5     /// <para>Определяет счётчик.</para>
6     /// </summary>
7     /// <typeparam name="TResult">
8     /// <para>The count result type.</para>
9     /// <para>Тип результата подсчёта.</para>
10    /// </typeparam>
11    public interface ICounter<out TResult>
12    {
13        /// <summary>
14        /// <para>Performs a count.</para>
15        /// <para>Выполняет подсчёт.</para>
16        /// </summary>
17        /// <returns>
18        /// <para>The count result.</para>
19        /// <para>Результат подсчёта.</para>
20        /// </returns>
21        TResult Count();
22    }
23 }
```

1.3 ./csharp/Platform.Interfaces/ICriterionMatcher.cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a criterion matcher, that contains a specific method for determining
5     /// ↪ whether the argument matches the criterion or not.</para>
6     /// <para>Определяет объект который проверяет соответствие критерию и содержит конкретный
7     /// ↪ метод для определения, соответствует ли аргумент критерию или нет.</para>
8     /// </summary>
9     /// <typeparam name="TArgument">
10    /// <para>Argument type.</para>
11    /// <para>Тип аргумента.</para>
12    /// </typeparam>
13    public interface ICriterionMatcher<in TArgument>
14    {
15        /// <summary>
16        /// <para>Determines whether the argument matches the criterion.</para>
17        /// <para>Определяет, соответствует ли аргумент критерию.</para>
18    }
```

```

16     /// </summary>
17     /// <param name="argument">
18     /// <para>The argument.</para>
19     /// <para>Аргумент.</para>
20     /// </param>
21     /// <returns>
22     /// <para>A value that determines whether the argument matches the criterion.</para>
23     /// <para>Значение, определяющее соответствует ли аргумент критерию.</para>
24     /// </returns>
25     bool IsMatched(TArgument argument);
26 }
27 }

```

1.4 ./csharp/Platform.Interfaces/IFactory.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a factory that produces instances of a specific type.</para>
5     /// <para>Определяет фабрику, которая производит экземпляры определенного типа.</para>
6     /// </summary>
7     /// <typeparam name="TProduct">
8     /// <para>Type of produced instances.</para>
9     /// <para>Тип производимых экземпляров.</para>
10    /// </typeparam>
11    public interface IFactory<out TProduct>
12    {
13        /// <summary>
14        /// <para>Creates an instance of <typeparamref name="TProduct"/> type.</para>
15        /// <para>Создает экземпляр типа <typeparamref name="TProduct"/>.</para>
16        /// </summary>
17        /// <returns>
18        /// <para>The instance of <typeparamref name="TProduct"/> type.</para>
19        /// <para>Экземпляр типа <typeparamref name="TProduct"/>.</para>
20        /// </returns>
21        TProduct Create();
22    }
23 }

```

1.5 ./csharp/Platform.Interfaces/IProperties.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a properties operator that is able to get or set values of properties of a
5     ↪ object of a specific type.</para>
6     /// <para>Определяет оператор свойств, который может получать или устанавливать значения
7     ↪ свойств объекта определенного типа.</para>
8     /// </summary>
9     /// <typeparam name="TObject">
10    /// <para>Object type.</para>
11    /// <para>Тип объекта.</para>
12    /// </typeparam>
13    /// <typeparam name="TProperty">
14    /// <para>Property reference type.</para>
15    /// <para>Тип ссылки на свойство.</para>
16    /// </typeparam>
17    /// <typeparam name="TValue">
18    /// <para>Property value type.</para>
19    /// <para>Тип значения свойства.</para>
20    /// </typeparam>
21    public interface IProperties<in TObject, in TProperty, TValue>
22    {
23        /// <summary>
24        /// <para>Gets the value of the property in the specified object.</para>
25        /// <para>Получает значение свойства в указанном объекте.</para>
26        /// </summary>
27        /// <param name="object">
28        /// <para>The object reference.</para>
29        /// <para>Ссылка на объект.</para>
30        /// </param>
31        /// <param name="property">
32        /// <para>The property reference.</para>
33        /// <para>Ссылка на свойство.</para>
34        /// </param>
35        /// <returns>
36        /// <para>The value of the property.</para>
37        /// <para>Значение свойства.</para>
38        /// </returns>
39    }
40 }

```

```

37         TValue GetValue(TObject @object, TProperty property);
38
39         /// <summary>
40         /// <para>Sets the value of a property in the specified object.</para>
41         /// <para>Устанавливает значение свойства в указанном объекте.</para>
42         /// </summary>
43         /// <param name="object">
44         /// <para>The object reference.</para>
45         /// <para>Ссылка на объект.</para>
46         /// </param>
47         /// <param name="property">
48         /// <para>The property reference.</para>
49         /// <para>Ссылка на свойство.</para>
50         /// </param>
51         /// <param name="value">
52         /// <para>The value.</para>
53         /// <para>Значение.</para>
54         /// </param>
55         void SetValue(TObject @object, TProperty property, TValue value);
56     }
57 }

```

1.6 ./csharp/Platform.Interfaces/IProperty.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a specific property operator that is able to get or set values of that
5     ↪ property.</para>
6     /// <para>Определяет оператор определённого свойства, который может получать или
7     ↪ устанавливать его значения.</para>
8     /// </summary>
9     /// <typeparam name="TObject">
10    /// <para>Object type.</para>
11    /// <para>Тип объекта.</para>
12    /// </typeparam>
13    /// <typeparam name="TValue">
14    /// <para>Property value type.</para>
15    /// <para>Тип значения свойства.</para>
16    /// </typeparam>
17    public interface IProperty<in TObject, TValue> : ISetter<TValue, TObject>, IProvider<TValue,
18    ↪ TObject>
19    {
20    }
21 }

```

1.7 ./csharp/Platform.Interfaces/IProvider[TProvided, TArgument].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines the provider of objects/values for which an argument must be
5     ↪ specified.</para>
6     /// <para>Определяет поставщика объектов/значений, для получения которых необходимо указать
7     ↪ аргумент.</para>
8     /// </summary>
9     /// <typeparam name="TProvided">
10    /// <para>Type of provided objects/values.</para>
11    /// <para>Тип предоставляемых объектов/значений.</para>
12    /// </typeparam>
13    /// <typeparam name="TArgument">
14    /// <para>Argument type.</para>
15    /// <para>Тип аргумента.</para>
16    /// </typeparam>
17    public interface IProvider<out TProvided, in TArgument>
18    {
19        /// <summary>
20        /// <para>Provides an object(s)/value(s).</para>
21        /// <para>Предоставляет объект(ы)/значение(я).</para>
22        /// </summary>
23        /// <param name="argument">
24        /// <para>The argument required to acquire the object(s)/value(s).</para>
25        /// <para>Аргумент, необходимый для получения объекта(ов)/значения(ий).</para>
26        /// </param>
27        /// <returns>
28        /// <para>The object(s)/value(s).</para>
29        /// <para>Объект(ы)/значение(я).</para>
30        /// </returns>
31        TProvided Get(TArgument argument);
32    }
33 }

```

```

30     }
31 }

```

1.8 ./csharp/Platform.Interfaces/IProvider[TProvided].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines the provider of objects/values.</para>
5     /// <para>Определяет поставщика объектов/значений.</para>
6     /// </summary>
7     /// <typeparam name="TProvided">
8     /// <para>Type of provided object/value.</para>
9     /// <para>Тип предоставляемого объекта/значения.</para>
10    /// </typeparam>
11    public interface IProvider<out TProvided>
12    {
13        /// <summary>
14        /// <para>Provides an object(s)/value(s).</para>
15        /// <para>Предоставляет объект(ы)/значение(я).</para>
16        /// </summary>
17        /// <returns>
18        /// <para>The object(s)/value(s).</para>
19        /// <para>Объект(ы)/значение(я).</para>
20        /// </returns>
21        TProvided Get();
22    }
23 }

```

1.9 ./csharp/Platform.Interfaces/ISetter[TValue, TArgument].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines an setter that requires an argument to set the passed value as a new
5     ↪ state.</para>
6     /// <para>Определяет установщик, которому для установки переданного значения в качестве
7     ↪ нового состояния требуется аргумент.</para>
8     /// </summary>
9     /// <typeparam name="TValue">
10    /// <para>Type of set value.</para>
11    /// <para>Тип устанавливаемого значения.</para>
12    /// </typeparam>
13    /// <typeparam name="TArgument">
14    /// <para>The argument type.</para>
15    /// <para>Тип аргумента.</para>
16    /// </typeparam>
17    public interface ISetter<in TValue, in TArgument>
18    {
19        /// <summary>
20        /// <para>Sets the value of a specific property in the specified object.</para>
21        /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
22        /// </summary>
23        /// <param name="argument">
24        /// <para>The argument.</para>
25        /// <para>Аргумент.</para>
26        /// </param>
27        /// <param name="value">
28        /// <para>The value.</para>
29        /// <para>Значение.</para>
30        /// </param>
31        void Set(TArgument argument, TValue value);
32    }
33 }

```

1.10 ./csharp/Platform.Interfaces/ISetter[TValue].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines an setter that sets the passed value as a new state.</para>
5     /// <para>Определяет установщик, который устанавливает переданное значение в качестве нового
6     ↪ состояния.</para>
7     /// </summary>
8     /// <typeparam name="TValue">
9     /// <para>Type of set value.</para>
10    /// <para>Тип устанавливаемого значения.</para>
11    /// </typeparam>
12    public interface ISetter<in TValue>
13    {
14        /// <summary>
15        /// <para>Sets the value of a specific property in the specified object.</para>
16        /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
17        /// </summary>
18        void Set(TValue value);
19    }
20 }

```

```

13     /// <summary>
14     /// <para>Sets the value of a specific property in the specified object.</para>
15     /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
16     /// </summary>
17     /// <param name="value">
18     /// <para>The value.</para>
19     /// <para>Значение.</para>
20     /// </param>
21     void Set(TValue value);
22 }
23 }

```

1.11 ./csharp/Platform.Interfaces.Tests/InterfacesTests.cs

```

1  using Xunit;
2
3  #pragma warning disable CS0168 // Variable is declared but never used
4  #pragma warning disable CS0219 // Variable is assigned but its value is never used
5
6  namespace Platform.Interfaces.Tests
7  {
8      /// <summary>
9      /// <para>
10     /// Represents the interfaces tests.
11     /// </para>
12     /// <para></para>
13     /// </summary>
14     public static class InterfacesTests
15     {
16         /// <summary>
17         /// <para>
18         /// Tests that build test.
19         /// </para>
20         /// <para></para>
21         /// </summary>
22         [Fact]
23         public static void BuildTest()
24         {
25             ICounter<int, int> c1 = null;
26             ICounter<int> c2 = null;
27             ICriterionMatcher<int> cm1 = null;
28             IFactory<int> f1 = null;
29             IProperties<int, int, int> p1 = null;
30             IProperty<int, int> p2 = null;
31             IProvider<int, int> p3 = null;
32             IProvider<int> p4 = null;
33             ISetter<int, int> s1 = null;
34             ISetter<int> s2 = null;
35         }
36     }
37 }

```

Index

- ./csharp/Platform.Interfaces.Tests/InterfacesTests.cs, 5
- ./csharp/Platform.Interfaces/ICounter[TResult, TArgument].cs, 1
- ./csharp/Platform.Interfaces/ICounter[TResult].cs, 1
- ./csharp/Platform.Interfaces/ICriterionMatcher.cs, 1
- ./csharp/Platform.Interfaces/IFactory.cs, 2
- ./csharp/Platform.Interfaces/IProperties.cs, 2
- ./csharp/Platform.Interfaces/IProperty.cs, 3
- ./csharp/Platform.Interfaces/IProvider[TProvided, TArgument].cs, 3
- ./csharp/Platform.Interfaces/IProvider[TProvided].cs, 4
- ./csharp/Platform.Interfaces/ISetter[TValue, TArgument].cs, 4
- ./csharp/Platform.Interfaces/ISetter[TValue].cs, 4