

LinksPlatform's Platform.Interfaces Class Library

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

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a counter that requires passing an argument to perform a count.</para>
5     /// <para>Определяет счетчик, который требует передачи аргумента для выполнения
6     /// подсчёта.</para>
7     /// </summary>
8     /// <typeparam name="TArgument"><para>The argument type.</para><para>Тип
9     /// аргумента.</para></typeparam>
10    /// <typeparam name="TResult"><para>The count result type.</para><para>Тип результата
11    /// подсчёта.</para></typeparam>
12    public interface ICounter<out TResult, in TArgument>
13    {
14        /// <summary>
15        /// <para>Performs a count that requires passing an argument.</para>
16        /// <para>Выполняет посчёт, для которого требуется передача аргумент.</para>
17        /// </summary>
18        /// <param name="argument"><para>The argument.</para><para>Аргумент.</para></param>
19        /// <returns><para>The count result.</para><para>Результат подсчёта.</para></returns>
20        TResult Count(TArgument argument);
21    }
22 }
```

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

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

1.3 ./Platform.Interfaces/ICriterionMatcher.cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a criterion matcher, that contains a specific method of determining
5     /// whether the argument matches criterion or not.</para>
6     /// <para>Определяет объект который проверяет соответствие критерию и содержит конкретный
7     /// метод определения, соответствует ли аргумент критерию или нет.</para>
8     /// </summary>
9     /// <typeparam name="TArgument"><para>Argument type.</para><para>Тип
10    /// аргумента.</para></typeparam>
11    public interface ICriterionMatcher<in TArgument>
12    {
13        /// <summary>
14        /// <para>Determines whether the argument matches the criterion.</para>
15        /// <para>Определяет, соответствует ли аргумент критерию.</para>
16        /// </summary>
17        /// <param name="argument"><para>The argument.</para><para>Аргумент.</para></param>
18        /// <returns><para>A value that determines whether the argument matches the
19        /// criterion.</para><para>Значение, определяющие соответствует ли аргумент
20        /// критерию.</para></returns>
21        bool IsMatched(TArgument argument);
22    }
23 }
```

1.4 ./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"><para>Type of produced instances.</para><para>Тип
    ↪ производимых экземпляров.</para></typeparam>
8     public interface IFactory<out TProduct>
9     {
10         /// <summary>
11         /// <para>Creates an instance of <typeparamref name="TProduct"/> type.</para>
12         /// <para>Создает экземпляр типа <typeparamref name="TProduct"/>.</para>
13         /// </summary>
14         /// <returns><para>The instance of <typeparamref name="TProduct"/>
    ↪ type.</para><para>Экземпляр типа <typeparamref name="TProduct"/>.</para></returns>
15         TProduct Create();
16     }
17 }

```

1.5 ./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
    ↪ object of a specific type.</para>
5         /// <para>Определяет оператор свойств, который может получать или устанавливать значения
    ↪ свойств объекта определенного типа.</para>
6         /// </summary>
7         /// <typeparam name="TObject"><para>Object type.</para><para>Тип объекта.</para></typeparam>
8         /// <typeparam name="TProperty"><para>Property reference type.</para><para>Тип ссылки на
    ↪ свойство.</para></typeparam>
9         /// <typeparam name="TValue"><para>Property value type.</para><para>Тип значения
    ↪ свойства.</para></typeparam>
10        public interface IProperties<in TObject, in TProperty, TValue>
11        {
12            /// <summary>
13            /// <para>Gets the value of the property in the specified object.</para>
14            /// <para>Получает значение свойства в указанном объекте.</para>
15            /// </summary>
16            /// <param name="object"><para>The object reference.</para><para>Ссылка на
    ↪ объект.</para></param>
17            /// <param name="property"><para>The property reference.</para><para>Ссылка на
    ↪ свойство.</para></param>
18            /// <returns><para>The value of the property.</para><para>Значение
    ↪ свойства.</para></returns>
19            TValue GetValue(TObject @object, TProperty property);
20
21            /// <summary>
22            /// <para>Sets the value of a property in the specified object.</para>
23            /// <para>Устанавливает значение свойства в указанном объекте.</para>
24            /// </summary>
25            /// <param name="object"><para>The object reference.</para><para>Ссылка на
    ↪ объект.</para></param>
26            /// <param name="property"><para>The property reference.</para><para>Ссылка на
    ↪ свойство.</para></param>
27            /// <param name="value"><para>The value.</para><para>Значение</para></param>
28            void SetValue(TObject @object, TProperty property, TValue value);
29        }
30 }

```

1.6 ./Platform.Interfaces/IProperty.cs

```

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

```

1.7 ./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     /// <para>Определяет поставщика объектов/значений, для получения которых необходимо указать
6     /// </summary>
7     /// <typeparam name="TProvided"><para>Type of provided objects/values.</para><para>Тип
8     /// <typeparam name="TArgument"><para>Argument type.</para><para>Тип
9     public interface IProvider<out TProvided, in TArgument>
10     {
11         /// <summary>
12         /// <para>Provides an object/value.</para>
13         /// <para>Предоставляет объект/значение.</para>
14         /// </summary>
15         /// <param name="argument"><para>The argument required to acquire the
16         /// <returns><para>The object/value.</para><para>Объект/значение.</para></returns>
17         TProvided Get(TArgument argument);
18     }
19 }
```

1.8 ./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"><para>Type of provided object/value.</para><para>Тип
8     public interface IProvider<out TProvided>
9     {
10         /// <summary>
11         /// <para>Provides an object/value.</para>
12         /// <para>Предоставляет объект/значение.</para>
13         /// </summary>
14         /// <returns><para>Object/value.</para><para>Объект/значение.</para></returns>
15         TProvided Get();
16     }
17 }
```

1.9 ./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     /// <para>Определяет установщик, которому для установки переданного значения в качестве
6     /// </summary>
7     /// <typeparam name="TValue"><para>Type of setted value.</para><para>Тип устанавливаемого
8     /// <typeparam name="TArgument"><para>The argument type.</para><para>Тип
9     public interface ISetter<in TValue, in TArgument>
10     {
11         /// <summary>
12         /// <para>Sets the value of a specific property in the specified object.</para>
13         /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
14         /// </summary>
15         /// <param name="argument"><para>The argument.</para><para>Аргумент.</para></param>
16         /// <param name="value"><para>The value.</para><para>Значение.</para></param>
17         void Set(TArgument argument, TValue value);
18     }
19 }
```

1.10 ./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
    ↪ состояния.</para>
6     /// </summary>
7     /// <typeparam name="TValue"><para>Type of setted value.</para><para>Тип устанавливаемого
    ↪ значения.</para></typeparam>
8     public interface ISetter<in TValue>
9     {
10         /// <summary>
11         /// <para>Sets the value of a specific property in the specified object.</para>
12         /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
13         /// </summary>
14         /// <param name="value"><para>The value.</para><para>Значение.</para></param>
15         void Set(TValue value);
16     }
17 }

```

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

```

1     using Xunit;
2
3     #pragma warning disable CS0168 // Variable is declared but never used
4
5     namespace Platform.Interfaces.Tests
6     {
7         public static class InterfacesTests
8         {
9             [Fact]
10            public static void BuildTest()
11            {
12                ICounter<int, int> c1;
13                ICounter<int> c2;
14                ICriterionMatcher<int> cm1;
15                IFactory<int> f1;
16                IProperties<int, int, int> p1;
17                IProperty<int, int> p2;
18                IProvider<int, int> p3;
19                IProvider<int> p4;
20                ISetter<int, int> s1;
21                ISetter<int> s2;
22            }
23        }
24    }

```

Index

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