

LinksPlatform's Platform.Interfaces Class Library

1.1 ./csharp/Platform.Interfaces/ICli.cs

```
1 namespace Platform.Interfaces;
2
3 /// <summary>
4 /// <para>Defines command line interfaces for command that interacts with an operating
  ↵ system.</para>
5 /// <para>Определяет интерфейс командной строки, для команды взаимодействующей с операционной
  ↵ системой.</para>
6 /// </summary>
7 public interface ICli
8 {
9     /// <summary>
10    /// <para>Runs a command.</para>
11    /// <para>Запускает команду.</para>
12    /// </summary>
13    /// <param name="args">
14    /// <para>Arguments for a command.</para>
15    /// <para>Аргументы для команды.</para>
16    /// </param>
17    /// <returns>
18    /// <para>Returns command's exit code.</para>
19    /// <para>Возвращает код выхода команды.</para>
20    /// </returns>
21    int Run(params string[] args);
22 }
```

1.2 ./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 }
```

1.3 ./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 }

```

1.4 ./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     /// <para>Определяет объект который проверяет соответствие критерию и содержит конкретный
6     /// <para>метод для определения, соответствует ли аргумент критерию или нет.</para>
7     /// </summary>
8     /// <typeparam name="TArgument">
9     /// <para>Argument type.</para>
10    /// <para>Тип аргумента.</para>
11    /// </typeparam>
12    public interface ICriterionMatcher<in TArgument>
13    {
14        /// <summary>
15        /// <para>Determines whether the argument matches the criterion.</para>
16        /// <para>Определяет, соответствует ли аргумент критерию.</para>
17        /// </summary>
18        /// <param name="argument">
19        /// <para>The argument.</para>
20        /// <para>Аргумент.</para>
21        /// </param>
22        /// <returns>
23        /// <para>A value that determines whether the argument matches the criterion.</para>
24        /// <para>Значение, определяющие соответствует ли аргумент критерию.</para>
25        /// </returns>
26        bool IsMatched(TArgument argument);
27    }
28 }

```

1.5 ./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 }

```

1.6 ./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     /// <para>Определяет оператор свойств, который может получать или устанавливать значения
6     /// <para>свойств объекта определенного типа.</para>
7     /// </summary>
8     /// <typeparam name="TObject">
9     /// <para>Object type.</para>
10    /// <para>Тип объекта.</para>
11    /// </typeparam>

```

```

11  /// <typeparam name="TProperty">
12  /// <para>Property reference type.</para>
13  /// <para>Тип ссылки на свойство.</para>
14  /// </typeparam>
15  /// <typeparam name="TValue">
16  /// <para>Property value type.</para>
17  /// <para>Тип значения свойства.</para>
18  /// </typeparam>
19  public interface IProperties<in TObject, in TProperty, TValue>
20  {
21      /// <summary>
22      /// <para>Gets the value of the property in the specified object.</para>
23      /// <para>Получает значение свойства в указанном объекте.</para>
24      /// </summary>
25      /// <param name="object">
26      /// <para>The object reference.</para>
27      /// <para>Ссылка на объект.</para>
28      /// </param>
29      /// <param name="property">
30      /// <para>The property reference.</para>
31      /// <para>Ссылка на свойство.</para>
32      /// </param>
33      /// <returns>
34      /// <para>The value of the property.</para>
35      /// <para>Значение свойства.</para>
36      /// </returns>
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  }

```

1.7 ./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  }

```

1.8 ./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>

```

```

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

```

1.9 ./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 }

```

1.10 ./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
    ↪ state.</para>
5     /// <para>Определяет установщик, которому для установки переданного значения в качестве
    ↪ нового состояния требуется аргумент.</para>
6     /// </summary>
7     /// <typeparam name="TValue">
8     /// <para>Type of set value.</para>
9     /// <para>Тип устанавливаемого значения.</para>
10    /// </typeparam>
11    /// <typeparam name="TArgument">
12    /// <para>The argument type.</para>
13    /// <para>Тип аргумента.</para>
14    /// </typeparam>
15    public interface ISetter<in TValue, in TArgument>
16    {
17        /// <summary>
18        /// <para>Sets the value of a specific property in the specified object.</para>
19        /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
20        /// </summary>

```

```

21     /// <param name="argument">
22     /// <para>The argument.</para>
23     /// <para>Аргумент.</para>
24     /// </param>
25     /// <param name="value">
26     /// <para>The value.</para>
27     /// <para>Значение.</para>
28     /// </param>
29     void Set(TArgument argument, TValue value);
30 }
31 }

```

1.11 ./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        /// <param name="value">
19        /// <para>The value.</para>
20        /// <para>Значение.</para>
21        /// </param>
22        void Set(TValue value);
23    }
24 }

```

1.12 ./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     public static class InterfacesTests
9     {
10        [Fact]
11        public static void BuildTest()
12        {
13            ICounter<int, int> c1 = null;
14            ICounter<int> c2 = null;
15            ICriterionMatcher<int> cm1 = null;
16            IFactory<int> f1 = null;
17            IProperties<int, int, int> p1 = null;
18            IProperty<int, int> p2 = null;
19            IProvider<int, int> p3 = null;
20            IProvider<int> p4 = null;
21            ISetter<int, int> s1 = null;
22            ISetter<int> s2 = null;
23        }
24    }
25 }

```

Index

- ./csharp/Platform.Interfaces.Tests/InterfacesTests.cs, 5
- ./csharp/Platform.Interfaces/ICli.cs, 1
- ./csharp/Platform.Interfaces/ICounter[TResult, TArgument].cs, 1
- ./csharp/Platform.Interfaces/ICounter[TResult].cs, 1
- ./csharp/Platform.Interfaces/ICriterionMatcher.cs, 2
- ./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, 5