

LinksPlatform's Platform.Interfaces Class Library

./IConverter[T].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines a converter between two values of the same type.
5     /// Определяет конвертер между двумя значениями одного типа.
6     /// </summary>
7     /// <typeparam name="T">Type of value to convert. Тип преобразуемого значения.</typeparam>
8     public interface IConverter<T> : IConverter<T, T>
9     {
10     }
11 }
```

./IConverter[TSource, TTarget].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines a converter between two types (TSource and TTarget).
5     /// Определяет конвертер между двумя типами (исходным TSource и целевым TTarget).
6     /// </summary>
7     /// <typeparam name="TSource">Source type of conversion. Исходный тип конверсии.</typeparam>
8     /// <typeparam name="TTarget">Target type of conversion. Целевой тип конверсии.</typeparam>
9     public interface IConverter<in TSource, out TTarget>
10     {
11         /// <summary>
12         /// Converts the value of the source type (TSource) to the value of the target type.
13         /// Конвертирует значение исходного типа (TSource) в значение целевого типа.
14         /// </summary>
15         /// <param name="source">The source type value (TSource). Значение исходного типа
16         ↪ (TSource).</param>
17         /// <returns>The value is converted to the target type (TTarget). Значение
18         ↪ ковертированное в целевой тип (TTarget).</returns>
19         TTarget Convert(TSource source);
20     }
21 }
```

./ICounter[TArgument, TResult].cs

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

./ICounter[TResult].cs

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

./ICriterionMatcher.cs

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

./IFactory.cs

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

./IIncrementer.cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines an incrementer that increments any number with a specific type.
5     /// Определяет инкрементер, который выполняет приращение любого числа определенного типа.
6     /// </summary>
7     /// <typeparam name="TNumber">Type of incremented number. Тип увеличиваемого
8     ///   ↳ числа.</typeparam>
9     public interface IIncrementer<TNumber>
10    {
11        /// <summary>
12        /// Increments the number by a specific value.
13        /// Увеличивает число на определённое значение.
14        /// </summary>
15        /// <param name="number">The number to be incremented.</param>
16        /// <returns>The incremented number. Увеличенное число.</returns>
17        TNumber Increment(TNumber number);
18    }
19 }
```

./IPropertiesOperator.cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines a properties operator that is able to get or set values of properties of a
5     ///   ↳ object of a specific type.
6     /// Определяет оператор свойств, который может получать или устанавливать значения свойств
7     ///   ↳ объекта определенного типа.
8     /// </summary>
9     /// <typeparam name="TObject">Object type. Тип объекта.</typeparam>
10    /// <typeparam name="TProperty">Property reference type. Тип ссылки на свойство.</typeparam>
11    /// <typeparam name="TValue">Property value type. Тип значения свойства.</typeparam>
12    public interface IPropertiesOperator<in TObject, in TProperty, TValue>
```

```

11 {
12     /// <summary>
13     /// Gets the value of the property in the specified object.
14     /// Получает значение свойства в указанном объекте.
15     /// </summary>
16     /// <param name="object">The object reference. Ссылка на объект.</param>
17     /// <param name="property">The property reference. Ссылка на свойство.</param>
18     /// <returns>The value of the property. Значение свойства.</returns>
19     TValue GetValue(TObject @object, TProperty property);
20
21     /// <summary>
22     /// Sets the value of a property in the specified object.
23     /// Устанавливает значение свойства в указанном объекте.
24     /// </summary>
25     /// <param name="object">The object reference. Ссылка на объект.</param>
26     /// <param name="property">The property reference. Ссылка на свойство.</param>
27     /// <param name="value">The value. Значение.</param>
28     void SetValue(TObject @object, TProperty property, TValue value);
29 }
30 }

```

./IPropertyOperator.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines a specific property operator that is able to get or set values of that property.
5     /// Определяет оператор определённого свойства, который может получать или устанавливать его
6     /// значения.
7     /// </summary>
8     /// <typeparam name="TObject">Object type. Тип объекта.</typeparam>
9     /// <typeparam name="TValue">Property value type. Тип значения свойства.</typeparam>
10    public interface IPropertyOperator<in TObject, TValue> : IProvider<TValue, TObject>
11    {
12        /// <summary>
13        /// Sets the value of a specific property in the specified object.
14        /// Устанавливает значение определённого свойства в указанном объекте.
15        /// </summary>
16        /// <param name="object">The object reference. Ссылка на объект.</param>
17        /// <param name="value">The value. Значение.</param>
18        void Set(TObject @object, TValue value);
19    }
20 }

```

./IProvider[TProvided].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines the provider of objects.
5     /// Определяет поставщика объектов.
6     /// </summary>
7     /// <typeparam name="TProvided">Type of provided object. Тип предоставляемого
8     /// объекта.</typeparam>
9    public interface IProvider<out TProvided>
10    {
11        /// <summary>
12        /// Provides an object.
13        /// </summary>
14        /// <returns>The provided object.</returns>
15        TProvided Get();
16    }
17 }

```

./IProvider[TProvided, TArgument].cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// Defines the provider of objects for which an argument must be specified.
5     /// Определяет поставщика объектов, для получения которых необходимо указать аргумент.
6     /// </summary>
7     /// <typeparam name="TProvided">Type of provided objects. Тип предоставляемых
8     /// объектов.</typeparam>
9     /// <typeparam name="TArgument">Argument type. Тип аргумента.</typeparam>
10    public interface IProvider<out TProvided, in TArgument>
11    {
12        /// <summary>
13        /// Provides an object.
14        /// </summary>

```

```
14      /// <param name="argument">The argument required to acquire the object. Аргумент,  
    ↳   необходимый для получения объекта.</param>  
15      /// <returns>The object.</returns>  
16      TProvided Get(TArgument argument);  
17  }  
18 }
```

Index

./IConverter[TSource, TTarget].cs, 1
./IConverter[T].cs, 1
./ICounter[TArgument, TResult].cs, 1
./ICounter[TResult].cs, 1
./ICriterionMatcher.cs, 1
./IFactory.cs, 2
./IIncrementer.cs, 2
./IPropertiesOperator.cs, 2
./IPropertyOperator.cs, 3
./IProvider[TProvided, TArgument].cs, 3
./IProvider[TProvided].cs, 3