```
LinksPlatform's Platform Setters Class Library
    ./csharp/Platform.Setters/SetterBase.cs
   using System.Runtime.CompilerServices;
   using Platform.Interfaces;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
   namespace Platform.Setters
6
        /// <summary>
       /// < \verb|para>Represents| a base implementation for an setter that allows you to set a passed
9
           value as the result value.</para>
       /// <para>Представляет базовую реализацию для установщика, который позволяет установить
10
       \rightarrow переданное ему значение в качестве результирующего значения.
11
       /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип
12
           результирующего значения.</para></typeparam>
        /// <remarks>
13
       /// Must be class, not struct (in order to persist access to Result property value).
14
       /// </remarks>
15
       public abstract class SetterBase<TResult> : ISetter<TResult>
16
17
            /// <summary>
18
            /// <para>Represents the result value.</para>
19
           /// <para>Представляет результирующие значение.</para>
20
            /// </summary>
           protected TResult _result;
22
23
            /// <summary>
24
           /// <para>Gets result value.</para>
25
            /// <para>Возвращает результирующее значение.</para>
            /// </summary>
27
           public TResult Result => _result;
29
            /// <summary>
            /// <para>Initializes a new instance of the SetterBase class.</para>
31
            /// <para>Инициализирует новый экземпляр класса SetterBase.</para>
32
            /// </summary>
33
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           protected SetterBase() { }
3.5
36
            /// <summary>
            /// <para>Initializes a new instance of the SetterBase class using the passed-in value
38

→ as the default result value.
            /// <para>Инициализирует новый экземпляр класса SetterBase, используя переданное
39
               значение в качестве результирующего по умолчанию. </para>
            /// </summary>
40
            /// <param name="defaultValue"><para>The default result
                value.</para><para>Pезультирующее значение по умолчанию.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
42
           protected SetterBase(TResult defaultValue) => _result = defaultValue;
43
44
            /// <summary>
45
            /// <para>Sets the passed value as the result.</para>
46
            /// <para>Устанавливает переданное значение в качестве результирующего.</para>
47
            /// </summary>
48
            /// <param name="value"><para>The result value.</para><para>Результирующее
49

→ значение.</para></param>

            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public void Set(TResult value) => _result = value;
51
       }
52
   }
53
    ./csharp/Platform.Setters/SetterExtensions.cs
   using System.Collections.Generic;
   namespace Platform. Setters
       public static class SetterExtensions
5
           public static TDecision SetFirstFromListAndReturnTrue<TResult, TDecision>(this
               Setter<TResult, TDecision> setter, IList<TResult> list)
                setter.Set(list[0]);
                return setter.TrueValue;
10
            }
11
```

12

```
public static TDecision SetFirstFromFirstListAndReturnTrue<TResult, TDecision>(this
13
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
14
                setter.Set(list1[0]);
                return setter.TrueValue;
16
           }
18
           public static TDecision SetSecondFromFirstListAndReturnTrue<TResult, TDecision>(this
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
20
                setter.Set(list1[1]):
21
               return setter.TrueValue;
           }
23
24
           public static TDecision SetThirdFromFirstListAndReturnTrue<TResult, TDecision>(this
25
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
            {
26
                setter.Set(list1[2]);
                return setter.TrueValue;
28
29
30
           public static TDecision SetFirstFromSecondListAndReturnTrue<TResult, TDecision>(this
3.1
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
               setter.Set(list2[0])
33
                return setter.TrueValue;
34
           }
36
           public static TDecision SetSecondFromSecondListAndReturnTrue<TResult, TDecision>(this
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
                setter.Set(list2[1])
39
40
                return setter.TrueValue;
           }
41
42
           public static TDecision SetThirdFromSecondListAndReturnTrue<TResult, TDecision>(this
43
               Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
44
                setter.Set(list2[2])
4.5
                return setter. True Value;
46
47
       }
   }
49
    ./csharp/Platform.Setters/Setter[TResult, TDecision].cs
   using System.Collections.Generic;
   using System.Runtime.CompilerServices;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
   namespace Platform.Setters
6
        /// <summary>
8
       /// <para>Represents implementation for an setter that allows you to set a passed value as
           the result value. This setter implementation has additional methods that, simultaneously
           with setting the result value, return <typeparamref name="TDecision"/> values indicating
           true or false.</para>
       /// <para>Представляет реализацию для установщика, который позволяет установить переданное
10
           ему значение в качестве результирующего значения. В этой реализации установщика есть
           дополнительные методы, которые одновременно с установкой результирующего значения
           возвращают значения типа <typeparamref name="TDecision"/>, обозначающие истину или
        \hookrightarrow
           ложь.</para>
       /// </summary>
11
       /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип
12
           результирующего значения.</para></typeparam>
       /// <typeparam name="TDecision"><para>The type of value which will be used to make the
        decision.</para>Тип значения на основе которого будет приниматься
           решение.</para></typeparam>
       public class Setter<TResult, TDecision> : SetterBase<TResult>
14
           public readonly TDecision TrueValue;
16
           public readonly TDecision FalseValue;
18
            /// <summary>
19
           /// <para>Initializes a new instance of the Setter class using the passed-in value as
20
               the default result value.</para>
            /// <para>Инициализирует новый экземпляр класса Setter, используя переданные значения
            🛶 trueValue, falseValue, defaultValue в качестве результирующего по умолчанию.</para>
```

```
/// </summary>
22
            /// <param name="defaultValue"><para>The default result
                value.</para>Peзультирующее значение по умолчанию.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
25
                : base(defaultValue)
26
                TrueValue = trueValue;
28
                FalseValue = falseValue;
29
            }
30
            /// <summary>
32
33
            /// <para>Gets result value.</para>
            /// <para>Возвращает результирующее значение.</para>
34
            /// </summary>
35
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
36
            public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
            → default) { }
38
            /// <summary>
39
            /// <para>Gets result value.</para>
40
            /// <para>Возвращает результирующее значение.</para>
41
            /// </summary>
42
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public Setter(TResult defaultValue) : base(defaultValue) { }
44
45
            /// <summary>
46
            /// <para>Gets result value.</para>
47
            /// <para>Возвращает результирующее значение.</para>
48
            /// </summary>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
50
            public Setter() { }
51
52
            /// <summary>
53
            /// <para>Gets result value.</para>
54
            /// <para>Возвращает результирующее значение.</para>
            /// </summary>
56
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
57
58
            public TDecision SetAndReturnTrue(TResult value)
59
                 _result = <mark>value</mark>;
60
                return TrueValue;
61
            }
62
63
            /// <summary>
64
            /// <para>Gets result value.</para>
65
            /// <para>Возвращает результирующее значение.</para>
            /// </summary>
67
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
68
69
            public TDecision SetAndReturnFalse(TResult value)
70
                 _result = value;
71
                return FalseValue;
72
            }
73
74
            /// <summary>
75
            /// <para>Gets result value.</para>
            /// <para>Возвращает результирующее значение.</para>
77
            /// </summary>
78
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
79
            public TDecision SetFirstAndReturnTrue(IList<TResult> list)
80
81
                result = list[0];
83
                return TrueValue;
84
85
            /// <summary>
86
            /// <para>Gets result value.</para>
            /// <para>Возвращает результирующее значение.</para>
88
            /// </summary>
89
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
90
            public TDecision SetFirstAndReturnFalse(IList<TResult> list)
92
                 result = list[0];
93
                return FalseValue;
94
            }
95
        }
96
```

97 }

```
./csharp/Platform.Setters/Setter[TResult].cs
   using System.Runtime.CompilerServices;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
   namespace Platform.Setters
6
        /// <summary>
        /// <para>
        /// Represents the setter.
9
        /// </para>
10
        /// <para></para>
11
        /// </summary>
12
        /// <seealso cref="Setter{TResult, bool}"/>
13
        public class Setter<TResult> : Setter<TResult, bool>
14
15
            /// <summary>
16
            /// <para>
17
            /// \bar{\text{Initializes}} a new <see cref="Setter"/> instance.
            /// </para>
19
            /// <para></para>
20
            /// </summary>
            /// <param name="defaultValue">
22
            /// <para>A default value.</para>
23
            /// <para></para>
^{24}
            /// </param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
26
            public Setter(TResult defaultValue) : base(true, false, defaultValue) { }
28
            /// <summary>
29
            /// <para>
30
            /// \bar{\text{Initializes}} a new <see cref="Setter"/> instance.
31
            /// </para>
32
            /// <para></para>
33
            /// </summary>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
35
            public Setter() : base(true, false) { }
36
        }
37
   }
38
    ./csharp/Platform.Setters.Tests/SetterTests.cs
   using Xunit;
   namespace Platform.Setters.Tests
3
   {
4
        public class SetterTests
5
6
            |Fact|
            public void ParameterlessConstructedSetterTest()
                Setter<int> setter = new Setter<int>();
10
                Assert.Equal(default, setter.Result);
            }
12
13
            [Fact]
14
            public void ConstructedWithDefaultValueSetterTest()
15
16
                Setter<int> setter = new Setter<int>(9);
                Assert.Equal(9, setter.Result);
18
            }
19
20
            [Fact]
21
            public void MethodsWithBooleanReturnTypeTest()
                Setter<int> setter = new Setter<int>()
24
                Assert.True(setter.SetAndReturnTrue(1));
25
                Assert.Equal(1, setter.Result);
26
                Assert.False(setter.SetAndReturnFalse(2));
27
                Assert.Equal(2, setter.Result);
28
                Assert.True(setter.SetFirstAndReturnTrue(new int[] { 3 }));
29
                Assert.Equal(3, setter.Result);
30
                Assert.False(setter.SetFirstAndReturnFalse(new int[] { 4 }));
31
                Assert.Equal(4, setter.Result);
32
            }
34
            [Fact]
35
            public void MethodsWithIntegerReturnTypeTest()
37
                Setter<int, int> setter = new Setter<int, int>(1, 0);
```

```
Assert.Equal(1, setter.SetAndReturnTrue(1));
Assert.Equal(1, setter.Result);
Assert.Equal(0, setter.SetAndReturnFalse(2));
Assert.Equal(2, setter.Result);
Assert.Equal(1, setter.SetFirstAndReturnTrue(new int[] { 3 }));
Assert.Equal(3, setter.Result);
Assert.Equal(3, setter.Result);
Assert.Equal(0, setter.SetFirstAndReturnFalse(new int[] { 4 }));
Assert.Equal(4, setter.Result);
Assert.Equal(4, setter.Result);
```

Index

- ./csharp/Platform.Setters.Tests/SetterTests.cs, 4
 ./csharp/Platform.Setters/SetterBase.cs, 1
 ./csharp/Platform.Setters/SetterExtensions.cs, 1
 ./csharp/Platform.Setters/Setter[TResult, TDecision].cs, 2
 ./csharp/Platform.Setters/Setter[TResult].cs, 3