

LinksPlatform's Platform.Setters Class Library

1.1 ./csharp/Platform.Setters/SetterBase.cs

```
1 using System.Runtime.CompilerServices;
2 using Platform.Interfaces;
3
4 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6 namespace Platform.Setters
7 {
8     /// <summary>
9     /// <para>Represents a base implementation for an setter that allows you to set a passed
10     /// ↪ value as the result value.</para>
11     /// <para>Представляет базовую реализацию для установщика, который позволяет установить
12     /// ↪ переданное ему значение в качестве результирующего значения.</para>
13     /// </summary>
14     /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип
15     /// ↪ результирующего значения.</para></typeparam>
16     /// <remarks>
17     /// Must be class, not struct (in order to persist access to Result property value).
18     /// </remarks>
19     public abstract class SetterBase<TResult> : ISetter<TResult>
20     {
21         /// <summary>
22         /// <para>Represents the result value.</para>
23         /// <para>Представляет результирующее значение.</para>
24         /// </summary>
25         protected TResult _result;
26
27         /// <summary>
28         /// <para>Gets result value.</para>
29         /// <para>Возвращает результирующее значение.</para>
30         /// </summary>
31         public TResult Result => _result;
32
33         /// <summary>
34         /// <para>Initializes a new instance of the SetterBase class.</para>
35         /// <para>Инициализирует новый экземпляр класса SetterBase.</para>
36         /// </summary>
37         [MethodImpl(MethodImplOptions.AggressiveInlining)]
38         protected SetterBase() { }
39
40         /// <summary>
41         /// <para>Initializes a new instance of the SetterBase class using the passed-in value
42         /// ↪ as the default result value.</para>
43         /// <para>Инициализирует новый экземпляр класса SetterBase, используя переданное
44         /// ↪ значение в качестве результирующего по умолчанию.</para>
45         /// </summary>
46         /// <param name="defaultValue"><para>The default result
47         /// ↪ value.</para><para>Результирующее значение по умолчанию.</para></param>
48         [MethodImpl(MethodImplOptions.AggressiveInlining)]
49         protected SetterBase(TResult defaultValue) => _result = defaultValue;
50
51         /// <summary>
52         /// <para>Sets the passed value as the result.</para>
53         /// <para>Устанавливает переданное значение в качестве результирующего.</para>
54         /// </summary>
55         /// <param name="value"><para>The result value.</para><para>Результирующее
56         /// ↪ значение.</para></param>
57         [MethodImpl(MethodImplOptions.AggressiveInlining)]
58         public void Set(TResult value) => _result = value;
59     }
60 }
```

1.2 ./csharp/Platform.Setters/SetterExtensions.cs

```
1 using System.Collections.Generic;
2
3 namespace Platform.Setters
4 {
5     public static class SetterExtensions
6     {
7         public static TDecision SetFirstFromListAndReturnTrue<TResult, TDecision>(this
8         ↪ Setter<TResult, TDecision> setter, IList<TResult> list)
9         {
10             setter.Set(list[0]);
11             return setter.TrueValue;
12         }
13     }
14 }
```

```

13     public static TDecision SetFirstFromFirstListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
14     {
15         setter.Set(list1[0]);
16         return setter.TrueValue;
17     }
18
19     public static TDecision SetSecondFromFirstListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
20     {
21         setter.Set(list1[1]);
22         return setter.TrueValue;
23     }
24
25     public static TDecision SetThirdFromFirstListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
26     {
27         setter.Set(list1[2]);
28         return setter.TrueValue;
29     }
30
31     public static TDecision SetFirstFromSecondListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
32     {
33         setter.Set(list2[0]);
34         return setter.TrueValue;
35     }
36
37     public static TDecision SetSecondFromSecondListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
38     {
39         setter.Set(list2[1]);
40         return setter.TrueValue;
41     }
42
43     public static TDecision SetThirdFromSecondListAndReturnTrue<TResult, TDecision>(this
    ↳ Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
44     {
45         setter.Set(list2[2]);
46         return setter.TrueValue;
47     }
48 }
49 }

```

1.3 ./csharp/Platform.Setters/Setter[TResult, TDecision].cs

```

1  using System.Collections.Generic;
2  using System.Runtime.CompilerServices;
3
4  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6  namespace Platform.Setters
7  {
8      /// <summary>
9      /// <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>
10     /// <para>Представляет реализацию для установщика, который позволяет установить переданное
    ↳ 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
    ↳ ложь.</para>
11     /// </summary>
12     /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип
    ↳ результирующего значения.</para></typeparam>
13     /// <typeparam name="TDecision"><para>The type of value which will be used to make the
    ↳ decision.</para><para>Тип значения на основе которого будет приниматься
    ↳ решение.</para></typeparam>
14     public class Setter<TResult, TDecision> : SetterBase<TResult>
15     {
16         public readonly TDecision TrueValue;
17         public readonly TDecision FalseValue;
18
19         /// <summary>
20         /// <para>Initializes a new instance of the Setter class using the passed-in value as
    ↳ the default result value.</para>
21         /// <para>Инициализирует новый экземпляр класса Setter, используя переданные значения
    ↳ trueValue, falseValue, defaultValue в качестве результирующего по умолчанию.</para>

```

```

22     /// </summary>
23     /// <param name="defaultValue"><para>The default result
    ↪ value.</para><para>Результирующее значение по умолчанию.</para></param>
24     [MethodImpl(MethodImplOptions.AggressiveInlining)]
25     public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
26         : base(defaultValue)
27     {
28         TrueValue = trueValue;
29         FalseValue = falseValue;
30     }
31
32     /// <summary>
33     /// <para>Gets result value.</para>
34     /// <para>Возвращает результирующее значение.</para>
35     /// </summary>
36     [MethodImpl(MethodImplOptions.AggressiveInlining)]
37     public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
    ↪ default) { }
38
39     /// <summary>
40     /// <para>Gets result value.</para>
41     /// <para>Возвращает результирующее значение.</para>
42     /// </summary>
43     [MethodImpl(MethodImplOptions.AggressiveInlining)]
44     public Setter(TResult defaultValue) : base(defaultValue) { }
45
46     /// <summary>
47     /// <para>Gets result value.</para>
48     /// <para>Возвращает результирующее значение.</para>
49     /// </summary>
50     [MethodImpl(MethodImplOptions.AggressiveInlining)]
51     public Setter() { }
52
53     /// <summary>
54     /// <para>Gets result value.</para>
55     /// <para>Возвращает результирующее значение.</para>
56     /// </summary>
57     [MethodImpl(MethodImplOptions.AggressiveInlining)]
58     public TDecision SetAndReturnTrue(TResult value)
59     {
60         _result = value;
61         return TrueValue;
62     }
63
64     /// <summary>
65     /// <para>Gets result value.</para>
66     /// <para>Возвращает результирующее значение.</para>
67     /// </summary>
68     [MethodImpl(MethodImplOptions.AggressiveInlining)]
69     public TDecision SetAndReturnFalse(TResult value)
70     {
71         _result = value;
72         return FalseValue;
73     }
74
75     /// <summary>
76     /// <para>Gets result value.</para>
77     /// <para>Возвращает результирующее значение.</para>
78     /// </summary>
79     [MethodImpl(MethodImplOptions.AggressiveInlining)]
80     public TDecision SetFirstAndReturnTrue(ICollection<TResult> list)
81     {
82         _result = list[0];
83         return TrueValue;
84     }
85
86     /// <summary>
87     /// <para>Gets result value.</para>
88     /// <para>Возвращает результирующее значение.</para>
89     /// </summary>
90     [MethodImpl(MethodImplOptions.AggressiveInlining)]
91     public TDecision SetFirstAndReturnFalse(ICollection<TResult> list)
92     {
93         _result = list[0];
94         return FalseValue;
95     }
96 }
97 }

```

1.4 ./csharp/Platform.Setters/Setter[TResult].cs

```
1 using System.Runtime.CompilerServices;
2
3 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5 namespace Platform.Setters
6 {
7     /// <summary>
8     /// <para>
9     /// Represents the setter.
10    /// </para>
11    /// <para></para>
12    /// </summary>
13    /// <seealso cref="Setter{TResult, bool}"/>
14    public class Setter<TResult> : Setter<TResult, bool>
15    {
16        /// <summary>
17        /// <para>
18        /// Initializes a new <see cref="Setter"/> instance.
19        /// </para>
20        /// <para></para>
21        /// </summary>
22        /// <param name="defaultValue">
23        /// <para>A default value.</para>
24        /// <para></para>
25        /// </param>
26        [MethodImpl(MethodImplOptions.AggressiveInlining)]
27        public Setter(TResult defaultValue) : base(true, false, defaultValue) { }
28
29        /// <summary>
30        /// <para>
31        /// Initializes a new <see cref="Setter"/> instance.
32        /// </para>
33        /// <para></para>
34        /// </summary>
35        [MethodImpl(MethodImplOptions.AggressiveInlining)]
36        public Setter() : base(true, false) { }
37    }
38 }
```

1.5 ./csharp/Platform.Setters.Tests/SetterTests.cs

```
1 using Xunit;
2
3 namespace Platform.Setters.Tests
4 {
5     public class SetterTests
6     {
7         [Fact]
8         public void ParameterlessConstructedSetterTest()
9         {
10             Setter<int> setter = new Setter<int>();
11             Assert.Equal(default, setter.Result);
12         }
13
14         [Fact]
15         public void ConstructedWithDefaultValueSetterTest()
16         {
17             Setter<int> setter = new Setter<int>(9);
18             Assert.Equal(9, setter.Result);
19         }
20
21         [Fact]
22         public void MethodsWithBooleanReturnTypeTest()
23         {
24             Setter<int> setter = new Setter<int>();
25             Assert.True(setter.SetAndReturnTrue(1));
26             Assert.Equal(1, setter.Result);
27             Assert.False(setter.SetAndReturnFalse(2));
28             Assert.Equal(2, setter.Result);
29             Assert.True(setter.SetFirstAndReturnTrue(new int[] { 3 }));
30             Assert.Equal(3, setter.Result);
31             Assert.False(setter.SetFirstAndReturnFalse(new int[] { 4 }));
32             Assert.Equal(4, setter.Result);
33         }
34
35         [Fact]
36         public void MethodsWithIntegerReturnTypeTest()
37         {
38             Setter<int, int> setter = new Setter<int, int>(1, 0);
```

```
39     Assert.Equal(1, setter.SetAndReturnTrue(1));
40     Assert.Equal(1, setter.Result);
41     Assert.Equal(0, setter.SetAndReturnFalse(2));
42     Assert.Equal(2, setter.Result);
43     Assert.Equal(1, setter.SetFirstAndReturnTrue(new int[] { 3 }));
44     Assert.Equal(3, setter.Result);
45     Assert.Equal(0, setter.SetFirstAndReturnFalse(new int[] { 4 }));
46     Assert.Equal(4, setter.Result);
47 }
48 }
49 }
```

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