

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/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
10     ↪ the result value. This setter implementation has additional methods that, simultaneously
11     ↪ with setting the result value, return <typeparamref name="TDecision"/> values indicating
12     ↪ true or false.</para>
13 }
```

```

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      /// <summary>
17      /// <para>
18      /// The true value.
19      /// </para>
20      /// <para></para>
21      /// </summary>
22      private readonly TDecision _trueValue;
23      /// <summary>
24      /// <para>
25      /// The false value.
26      /// </para>
27      /// <para></para>
28      /// </summary>
29      private readonly TDecision _falseValue;
30
31      /// <summary>
32      /// <para>Initializes a new instance of the Setter class using the passed-in value as
    ↳ the default result value.</para>
33      /// <para>Инициализирует новый экземпляр класса Setter, используя переданные значения
    ↳ trueValue, falseValue, defaultValue в качестве результирующего по умолчанию.</para>
34      /// </summary>
35      /// <param name="defaultValue"><para>The default result
    ↳ value.</para><para>Результирующее значение по умолчанию.</para></param>
36      [MethodImpl(MethodImplOptions.AggressiveInlining)]
37      public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
38          : base(defaultValue)
39      {
40          _trueValue = trueValue;
41          _falseValue = falseValue;
42      }
43
44      /// <summary>
45      /// <para>Gets result value.</para>
46      /// <para>Возвращает результирующее значение.</para>
47      /// </summary>
48      [MethodImpl(MethodImplOptions.AggressiveInlining)]
49      public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
    ↳ default) { }
50
51      /// <summary>
52      /// <para>Gets result value.</para>
53      /// <para>Возвращает результирующее значение.</para>
54      /// </summary>
55      [MethodImpl(MethodImplOptions.AggressiveInlining)]
56      public Setter(TResult defaultValue) : base(defaultValue) { }
57
58      /// <summary>
59      /// <para>Gets result value.</para>
60      /// <para>Возвращает результирующее значение.</para>
61      /// </summary>
62      [MethodImpl(MethodImplOptions.AggressiveInlining)]
63      public Setter() { }
64
65      /// <summary>
66      /// <para>Gets result value.</para>
67      /// <para>Возвращает результирующее значение.</para>
68      /// </summary>
69      [MethodImpl(MethodImplOptions.AggressiveInlining)]
70      public TDecision SetAndReturnTrue(TResult value)
71      {
72          _result = value;
73          return _trueValue;
74      }
75
76      /// <summary>

```

```

77     /// <para>Gets result value.</para>
78     /// <para>Возвращает результирующее значение.</para>
79     /// </summary>
80     [MethodImpl(MethodImplOptions.AggressiveInlining)]
81     public TDecision SetAndReturnFalse(TResult value)
82     {
83         _result = value;
84         return _falseValue;
85     }
86
87     /// <summary>
88     /// <para>Gets result value.</para>
89     /// <para>Возвращает результирующее значение.</para>
90     /// </summary>
91     [MethodImpl(MethodImplOptions.AggressiveInlining)]
92     public TDecision SetFirstAndReturnTrue(IList<TResult> list)
93     {
94         _result = list[0];
95         return _trueValue;
96     }
97
98     /// <summary>
99     /// <para>Gets result value.</para>
100    /// <para>Возвращает результирующее значение.</para>
101    /// </summary>
102    [MethodImpl(MethodImplOptions.AggressiveInlining)]
103    public TDecision SetFirstAndReturnFalse(IList<TResult> list)
104    {
105        _result = list[0];
106        return _falseValue;
107    }
108 }
109 }

```

1.3 ./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 }

```

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

```

1  using Xunit;
2
3  namespace Platform.Setters.Tests
4  {

```

```

5  /// <summary>
6  /// <para>
7  /// Represents the setter tests.
8  /// </para>
9  /// <para></para>
10 /// </summary>
11 public class SetterTests
12 {
13     /// <summary>
14     /// <para>
15     /// Tests that parameterless constructed setter test.
16     /// </para>
17     /// <para></para>
18     /// </summary>
19     [Fact]
20     public void ParameterlessConstructedSetterTest()
21     {
22         Setter<int> setter = new Setter<int>();
23         Assert.Equal(default, setter.Result);
24     }
25
26     /// <summary>
27     /// <para>
28     /// Tests that constructed with default value setter test.
29     /// </para>
30     /// <para></para>
31     /// </summary>
32     [Fact]
33     public void ConstructedWithDefaultValueSetterTest()
34     {
35         Setter<int> setter = new Setter<int>(9);
36         Assert.Equal(9, setter.Result);
37     }
38
39     /// <summary>
40     /// <para>
41     /// Tests that methods with boolean return type test.
42     /// </para>
43     /// <para></para>
44     /// </summary>
45     [Fact]
46     public void MethodsWithBooleanReturnTypeTest()
47     {
48         Setter<int> setter = new Setter<int>();
49         Assert.True(setter.SetAndReturnTrue(1));
50         Assert.Equal(1, setter.Result);
51         Assert.False(setter.SetAndReturnFalse(2));
52         Assert.Equal(2, setter.Result);
53         Assert.True(setter.SetFirstAndReturnTrue(new int[] { 3 }));
54         Assert.Equal(3, setter.Result);
55         Assert.False(setter.SetFirstAndReturnFalse(new int[] { 4 }));
56         Assert.Equal(4, setter.Result);
57     }
58
59     /// <summary>
60     /// <para>
61     /// Tests that methods with integer return type test.
62     /// </para>
63     /// <para></para>
64     /// </summary>
65     [Fact]
66     public void MethodsWithIntegerReturnTypeTest()
67     {
68         Setter<int, int> setter = new Setter<int, int>(1, 0);
69         Assert.Equal(1, setter.SetAndReturnTrue(1));
70         Assert.Equal(1, setter.Result);
71         Assert.Equal(0, setter.SetAndReturnFalse(2));
72         Assert.Equal(2, setter.Result);
73         Assert.Equal(1, setter.SetFirstAndReturnTrue(new int[] { 3 }));
74         Assert.Equal(3, setter.Result);
75         Assert.Equal(0, setter.SetFirstAndReturnFalse(new int[] { 4 }));
76         Assert.Equal(4, setter.Result);
77     }
78 }
79 }

```

Index

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