

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 range between minumum and maximum values.</para>
10    /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
11    /// </summary>
12    /// <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 a range between minumum and maximum values.</para>
19        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
20        /// </summary>
21        /// <remarks>
22        /// Must be class, not struct (in order to persist access to Result property value).
23        /// </remarks>
24        protected TResult _result;
25
26        /// <summary>
27        /// <para>Represents a range between minumum and maximum values.</para>
28        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
29        /// </summary>
30        /// <remarks>
31        /// Must be class, not struct (in order to persist access to Result property value).
32        /// </remarks>
33        public TResult Result => _result;
34
35        /// <summary>
36        /// <para>Represents a range between minumum and maximum values.</para>
37        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
38        /// </summary>
39        /// <remarks>
40        /// Must be class, not struct (in order to persist access to Result property value).
41        /// </remarks>
42        [MethodImpl(MethodImplOptions.AggressiveInlining)]
43        protected SetterBase() { }
44
45        /// <summary>
46        /// <para>Represents a range between minumum and maximum values.</para>
47        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
48        /// </summary>
49        /// <remarks>
50        /// Must be class, not struct (in order to persist access to Result property value).
51        /// </remarks>
52        [MethodImpl(MethodImplOptions.AggressiveInlining)]
53        protected SetterBase(TResult defaultValue) => _result = defaultValue;
54
55        /// <summary>
56        /// <para>Represents a range between minumum and maximum values.</para>
57        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
58        /// </summary>
59        /// <remarks>
60        /// Must be class, not struct (in order to persist access to Result property value).
61        /// </remarks>
62        [MethodImpl(MethodImplOptions.AggressiveInlining)]
63        public void Set(TResult value) => _result = value;
64    }
65 }
```

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     public class Setter<TResult, TDecision> : SetterBase<TResult>
9     {
10         private readonly TDecision _trueValue;
```

```

11     private readonly TDecision _falseValue;
12
13     [MethodImpl(MethodImplOptions.AggressiveInlining)]
14     public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
15         : base(defaultValue)
16     {
17         _trueValue = trueValue;
18         _falseValue = falseValue;
19     }
20
21     [MethodImpl(MethodImplOptions.AggressiveInlining)]
22     public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
23         ↪ default) { }
24
25     [MethodImpl(MethodImplOptions.AggressiveInlining)]
26     public Setter(TResult defaultValue) : base(defaultValue) { }
27
28     [MethodImpl(MethodImplOptions.AggressiveInlining)]
29     public Setter() { }
30
31     [MethodImpl(MethodImplOptions.AggressiveInlining)]
32     public TDecision SetAndReturnTrue(TResult value)
33     {
34         _result = value;
35         return _trueValue;
36     }
37
38     [MethodImpl(MethodImplOptions.AggressiveInlining)]
39     public TDecision SetAndReturnFalse(TResult value)
40     {
41         _result = value;
42         return _falseValue;
43     }
44
45     [MethodImpl(MethodImplOptions.AggressiveInlining)]
46     public TDecision SetFirstAndReturnTrue(ICollection<TResult> list)
47     {
48         _result = list[0];
49         return _trueValue;
50     }
51
52     [MethodImpl(MethodImplOptions.AggressiveInlining)]
53     public TDecision SetFirstAndReturnFalse(ICollection<TResult> list)
54     {
55         _result = list[0];
56         return _falseValue;
57     }
58 }

```

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     public class Setter<TResult> : Setter<TResult, bool>
8     {
9         [MethodImpl(MethodImplOptions.AggressiveInlining)]
10        public Setter(TResult defaultValue) : base(true, false, defaultValue) { }
11
12        [MethodImpl(MethodImplOptions.AggressiveInlining)]
13        public Setter() : base(true, false) { }
14    }
15 }

```

1.4 ./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 }

```

```

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`, 2
`./csharp/Platform.Setters/SetterBase.cs`, 1
`./csharp/Platform.Setters/Setter[TResult, TDecision].cs`, 1
`./csharp/Platform.Setters/Setter[TResult].cs`, 2