LinksPlatform's Platform.Setters Class Library

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

```csharp
using System.Runtime.CompilerServices;
using Platform.Interfaces;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Setters
{
    /// <summary>
    /// <para>Provides a base implementation for an setter that allows you to set a passed value
    ///    as the resulting value.</para>
    /// <para>Предоставляет базовую реализацию для установщика, который позволяет установить
    ///    переданное значение в качестве результирующего значения.</para>
    /// </summary>
    /// <remarks>
    /// Must be class, not struct (in order to persist access to Result property value).
    /// </remarks>
    public abstract class SetterBase<TResult> : ISetter<TResult>
    {
        /// <summary>
        /// <para>Represents a range between minumum and maximum values.</para>
        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
        /// </summary>
        protected TResult _result;

        /// <summary>
        /// <para>Represents a range between minumum and maximum values.</para>
        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
        /// </summary>
        public TResult Result => _result;

        /// <summary>
        /// <para>Represents a range between minumum and maximum values.</para>
        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        protected SetterBase() { }

        /// <summary>
        /// <para>Sets the passed-in default value as the result.</para>
        /// <para>Устанавливает переданное значение по умолчанию в качестве
        ///    результирующего.</para>
        /// </summary>
        /// <param name="defaultValue"><para>The default result
        ///    value.</para><para>Результирующее значение по умолчанию.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        protected SetterBase(TResult defaultValue) => _result = defaultValue;

        /// <summary>
        /// <para>Sets the passed value as the result.</para>
        /// <para>Устанавливает переданное значение в качестве результирующего.</para>
        /// </summary>
        /// <param name="value"><para>The result value.</para><para>Результирующее
        ///    значение.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public void Set(TResult value) => _result = value;
    }
}
```

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

```csharp
using System.Collections.Generic;
using System.Runtime.CompilerServices;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Setters
{
    public class Setter<TResult, TDecision> : SetterBase<TResult>
    {
        private readonly TDecision _trueValue;
        private readonly TDecision _falseValue;

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
            : base(defaultValue)
        {
            _trueValue = trueValue;
            _falseValue = falseValue;
```

```
19              }
20
21              [MethodImpl(MethodImplOptions.AggressiveInlining)]
22              public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
   ↪   default) { }
23
24              [MethodImpl(MethodImplOptions.AggressiveInlining)]
25              public Setter(TResult defaultValue) : base(defaultValue) { }
26
27              [MethodImpl(MethodImplOptions.AggressiveInlining)]
28              public Setter() { }
29
30              [MethodImpl(MethodImplOptions.AggressiveInlining)]
31              public TDecision SetAndReturnTrue(TResult value)
32              {
33                  _result = value;
34                  return _trueValue;
35              }
36
37              [MethodImpl(MethodImplOptions.AggressiveInlining)]
38              public TDecision SetAndReturnFalse(TResult value)
39              {
40                  _result = value;
41                  return _falseValue;
42              }
43
44              [MethodImpl(MethodImplOptions.AggressiveInlining)]
45              public TDecision SetFirstAndReturnTrue(IList<TResult> list)
46              {
47                  _result = list[0];
48                  return _trueValue;
49              }
50
51              [MethodImpl(MethodImplOptions.AggressiveInlining)]
52              public TDecision SetFirstAndReturnFalse(IList<TResult> list)
53              {
54                  _result = list[0];
55                  return _falseValue;
56              }
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          [Fact]
15          public void ConstructedWithDefaultValueSetterTest()
16          {
17              Setter<int> setter = new Setter<int>(9);
18              Assert.Equal(9, setter.Result);
19          }
```

```csharp
        [Fact]
        public void MethodsWithBooleanReturnTypeTest()
        {
            Setter<int> setter = new Setter<int>();
            Assert.True(setter.SetAndReturnTrue(1));
            Assert.Equal(1, setter.Result);
            Assert.False(setter.SetAndReturnFalse(2));
            Assert.Equal(2, setter.Result);
            Assert.True(setter.SetFirstAndReturnTrue(new int[] { 3 }));
            Assert.Equal(3, setter.Result);
            Assert.False(setter.SetFirstAndReturnFalse(new int[] { 4 }));
            Assert.Equal(4, setter.Result);
        }

        [Fact]
        public void MethodsWithIntegerReturnTypeTest()
        {
            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(0, setter.SetFirstAndReturnFalse(new int[] { 4 }));
            Assert.Equal(4, setter.Result);
        }
    }
}
```

Index