LinksPlatform's Platform.Setters Class Library

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

```
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>Represents a range between minumum and maximum values.</para>
    /// <para>Представляет диапазон между минимальным и максимальным значениями.</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>Represents a range between minumum and maximum values.</para>
        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        protected SetterBase(TResult defaultValue) => _result = defaultValue;

        /// <summary>
        /// <para>Represents a range between minumum and maximum values.</para>
        /// <para>Представляет диапазон между минимальным и максимальным значениями.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public void Set(TResult value) => _result = value;
    }
}
```

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

```
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;
        }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
        ↪  default) { }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TResult defaultValue) : base(defaultValue) { }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter() { }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetAndReturnTrue(TResult value)
        {
            _result = value;
            return _trueValue;
        }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetAndReturnFalse(TResult value)
        {
            _result = value;
            return _falseValue;
        }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetFirstAndReturnTrue(IList<TResult> list)
        {
            _result = list[0];
            return _trueValue;
        }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetFirstAndReturnFalse(IList<TResult> list)
        {
            _result = list[0];
            return _falseValue;
        }
    }
}
```

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

```
using System.Runtime.CompilerServices;

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

namespace Platform.Setters
{
    public class Setter<TResult> : Setter<TResult, bool>
    {
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TResult defaultValue) : base(true, false, defaultValue) { }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter() : base(true, false) { }
    }
}
```

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

```
using Xunit;

namespace Platform.Setters.Tests
{
    public class SetterTests
    {
        [Fact]
        public void ParameterlessConstructedSetterTest()
        {
            Setter<int> setter = new Setter<int>();
            Assert.Equal(default, setter.Result);
        }

        [Fact]
        public void ConstructedWithDefaultValueSetterTest()
        {
            Setter<int> setter = new Setter<int>(9);
            Assert.Equal(9, setter.Result);
        }

        [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