LinksPlatform's Platform.Diagnostics Class Library

1.1 ./csharp/Platform.Diagnostics/Performance.cs

```csharp
using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;

namespace Platform.Diagnostics
{
    /// <summary>
    /// <para>Provides a set of methods for performance measurement.</para>
    /// <para>Предоставляет набор методов для замера производительности.</para>
    /// </summary>
    public static class Performance
    {
        /// <summary>
        /// <para>Measures the time it takes to execute the action by executing it and using
        ///   <see cref="Stopwatch"/>.</para>
        /// <para>Измеряет время, которое занимает выполнение действия, выполняя его и используя
        ///   <see cref="Stopwatch"/>.</para>
        /// </summary>
        /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делегат
        ///   <see cref="Action"/>.</para></param>
        /// <returns>
        /// <para>The time interval representing the time it took to execute the action.</para>
        /// <para>Временной интервал, представляющий время, которое потребовалось для выполнения
        ///   действия.</para>
        /// </returns>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public static TimeSpan Measure(Action action)
        {
            var sw = Stopwatch.StartNew();
            action();
            sw.Stop();
            return sw.Elapsed;
        }
    }
}
```

1.2 ./csharp/Platform.Diagnostics.Tests/PerformanceTests.cs

```csharp
using System;
using Xunit;

namespace Platform.Diagnostics.Tests
{
    /// <summary>
    /// <para>
    /// Represents the performance tests.
    /// </para>
    /// <para></para>
    /// </summary>
    public class PerformanceTests
    {
        /// <summary>
        /// <para>
        /// Tests that measure test.
        /// </para>
        /// <para></para>
        /// </summary>
        [Fact]
        public void MeasureTest()
        {
            var result = 0;
            var measurement = Performance.Measure(() => result = 1 + 1);
            Assert.Equal(2, result);
            Assert.True(measurement > TimeSpan.Zero);
        }
    }
}
```

Index