c# check if a timespan range is between timespan range and how many hours

Ask Question

2

1

Assuming I have 3 time ranges:

```
07:00 - 18:00
18:00 - 23:00
23:00 - 07:00
```

and the code:

```
public class TimeShift
{
   public TimeSpan Start { get; set; }
   public TimeSpan End { get; set; }
}

List<TimeShift> shifts = new List<TimeShift>();
```

How can I check if every item in the list is between the 3 ranges above and how many hours?

For example one `TimeShift` where:

```
Start: 07:00
End:   23:30
```

then that means 16.5 hours.

For the examples above:

```
Range 1: 11 hours
Range 2: 5 hours
Range 3: 0.5 hours
```

c# datetime timespan date-range

edited Dec 2 '16 at 1:50

Nkosi
120k 17 142 206

asked Dec 1 '16 at 23:43

user2818430
1,908 10 46 101

`.TotalHours` is available on `TimeSpan`. However, for shifts, I'd recommend using `DateTime` rather than `TimeSpan`. Or, a combination of `DateTime StartTime` and `TimeSpan Duration` – Rob ♦ Dec 2 '16 at 0:06

5 Answers

2

Here is a solution including tests:

Calc

```
public class TimeSpacCalculator
{
    public static TimeSpan GetTimeSpanIntersect(TimeShift input, Tim
TimeSpan end)
    {
        // Loopsback input from 23:59 - 00:00
        if (input.Start > input.End)
            return GetTimeSpanIntersect(new TimeShift(input.Start,
```

```
TimeSpan.FromHours(24)), start, end) +
                    GetTimeSpanIntersect(new TimeShift(TimeSpan.FromH
start, end);

        // Loopsback Shift from 23:59 - 00:00
        if (start > end)
            return GetTimeSpanIntersect(input, new TimeSpan(), end)
                   GetTimeSpanIntersect(input, start, TimeSpan.FromH
        if (input.End < start)
            return new TimeSpan();

        if (input.Start > end)
            return new TimeSpan();

        var actualStart = input.Start < start
            ? start
            : input.Start;

        var actualEnd = input.End > end
            ? end
            : input.End;

        return actualEnd - actualStart;
    }
}
```

Classes

```
public class TimeRange : TimeShift
{
    public TimeRange(string name, TimeSpan start, TimeSpan end) : ba
    {
        Name = name;
    }

    public string Name { get; set; }
}

public class TimeShift
{
    public TimeShift(TimeSpan start, TimeSpan end)
    {
        Start = start;
        End = end;
    }
```

```
    public TimeSpan Start { get; set; }
    public TimeSpan End { get; set; }
}
```

Tests

```
[TestFixture]
internal class TimShiftTests
{
    [Test]
    [TestCase(7, 23.5, 11, 5, 0.5)]
    [TestCase(22, 7.5, 0.5, 1, 8)]
    public void Test(double inputStartHours, double inputEndHours, d
expectedRange1Hours, double expectedRange2Hours, double expectedRang
    {
        var input = new TimeShift(TimeSpan.FromHours(inputStartHours
TimeSpan.FromHours(inputEndHours));

        var ranges = new List<TimeRange>
        {
            new TimeRange("Range1", TimeSpan.FromHours(7), TimeSpan.
            new TimeRange("Range2", TimeSpan.FromHours(18), TimeSpar
            new TimeRange("Range3", TimeSpan.FromHours(23), TimeSpar
        };


        var result = new Dictionary<string, TimeSpan>();

        foreach (var range in ranges)
        {
            var time = TimeSpacCalculator.GetTimeSpanIntersect(input
range.End);

            result.Add(range.Name, time);

            Console.WriteLine($"{range.Name}: " + time.TotalHours);
        }

        result["Range1"].Should().Be(TimeSpan.FromHours(expectedRang
        result["Range2"].Should().Be(TimeSpan.FromHours(expectedRang
        result["Range3"].Should().Be(TimeSpan.FromHours(expectedRang
}
```

edited Dec 2 '16 at 20:39

answered Dec 2 '16 at 0:28

Michal Ciechan
8,708 10 56 96

Good answer. Had to do something similar recently. Take a look at how I resolved it. stackoverflow.com/a/40923461/5233410 – Nkosi Dec 2 '16 at 1:46

If my shift starts from 22:00 and ends at 07:30 the next day, the result will be 0 for all. How can I resolve that? – active92 Dec 2 '16 at 1:48

1 @active92 Fixed the `GetTimeSpanIntersect` and updated Test to use TestCases and include your test case. – Michal Ciechan Dec 2 '16 at 20:40

@MichalCiechan got it working. cheers. – active92 Dec 3 '16 at 5:25

0

I think I found the solution:

```
private double GetHours(TimeSpan start, TimeSpan end, TimeSpan start
endTarget)
{
   double result = 0;
   if (startTarget >= start && endTarget <= end)
   {
      result = (endTarget - startTarget).TotalHours;
   }

   if ((startTarget >= start && startTarget < end) && endTarget > en
   {
      result = (end - startTarget).TotalHours;
   }

   if (startTarget < start && (endTarget > start && endTarget <= end
   {
```

```
            result = (endTarget - start).TotalHours;
        }

        if (startTarget < start && endTarget > end)
        {
            result = (end - start).TotalHours;
        }
        return result;
    }
```

edited Dec 2 '16 at 10:26

Bojan B
1,816 4 13 22

answered Dec 2 '16 at 0:23

user2818430
1,908 10 46 101

Had to do something similar recently. Take a look at how I resolved it. stackoverflow.com/a/40923461/5233410 – Nkosi Dec 2 '16 at 1:47

0

Had a similar requirement for a recent project. Here is my experience in solving the same issue.

Given the requirements, the following classes were derived.

```
public interface IRange<T> : IEquatable<T> where T : IComparable {
    T Maximum { get; }
    T Minimum { get; }
}

public sealed class Range<T> : IRange<T>
    where T : IComparable {
    public Range(T minimum, T maximum) {
        Minimum = minimum;
        Maximum = maximum;
    }
```

```
    public T Maximum { get; private set; }

    public T Minimum { get; private set; }

    public override string ToString() {
        return string.Format("{{{0} - {1}}}", Minimum, Maximum);
    }

    public override int GetHashCode() {
        return ToString().GetHashCode();
    }

    public override bool Equals(object obj) {
        if (ReferenceEquals(null, obj)) {
            return false;
        }
        return obj is Range<T> && Equals((Range<T>)obj);
    }

    public bool Equals(T other) {
        return object.Equals(this.ToString(), other.ToString());
    }

}
```

supported by the following extension methods

```
public static class Range {
    /// <summary>
    /// Create an <seealso cref="IRange&lt;T&gt;"/> using the provid
maximum value
    /// </summary>
    public static IRange<T> Of<T>(T min, T max) where T : IComparabl
        return new Range<T>(min, max);
    }
    /// <summary>
    ///
    /// </summary>
    public static bool Contains<T>(this IRange<T> range, T value) wh
{
        return range.Minimum.CompareTo(value) <= 0 && value.CompareT
0;
    }
    /// <summary>
    ///
```

```
    /// </summary>
    public static bool IsOverlapped<T>(this IRange<T> range, IRange<
inclusive = false) where T : IComparable {
        return inclusive
            ? range.Minimum.CompareTo(other.Maximum) <= 0 &&
other.Minimum.CompareTo(range.Maximum) <= 0
            : range.Minimum.CompareTo(other.Maximum) < 0 &&
other.Minimum.CompareTo(range.Maximum) < 0;
    }
    /// <summary>
    ///
    /// </summary>
    public static IRange<T> GetIntersection<T>(this IRange<T> range,
bool inclusive = false) where T : IComparable {
        var start = new[] { range.Minimum, other.Minimum }.Max();
        var end = new[] { range.Maximum, other.Maximum }.Min();

        var valid = inclusive ? start.CompareTo(end) < 0 : start.Com

        return valid ? new Range<T>(start, end) : null;
    }
}
```

Here is a test adapted to your particular requirements

```
[TestClass]
public class TimeShiftTests : MiscUnitTests {
    [TestMethod]
    public void TimeShiftDurationTest() {
        var shifts = new List<string>(){
            "07:00 - 18:00",
            "18:00 - 23:00",
            "23:00 - 07:00"
        }.Select(s => ParseShift(s));

        var timeShift = "07:00 - 23:30";
        var totalExpectedHours = 16.5;
        var input = ParseShift(timeShift);

        var intersections = shifts
            .Select(shift => shift.GetIntersection(input))
            .ToArray();

        intersections.Length.Should().Be(3);

        var actualHours = intersections.Select(range => (range.Maxim
```

```
range.Minimum).TotalHours).ToArray();

        var totalActualHours = actualHours.Sum();

        totalActualHours.Should().Be(totalExpectedHours);

        actualHours[0].Should().Be(11);
        actualHours[1].Should().Be(5);
        actualHours[2].Should().Be(0.5);

    }

    private IRange<DateTime> ParseShift(string period, string format
        var tokens = period
            .Split(new[] { "to", "-" }, StringSplitOptions.RemoveEmp
            .Select(s => s.Trim().Replace(" ", string.Empty))
            .ToArray();

        if (tokens.Length != 2) throw new FormatException("time peri
formatted");

        var startDate = DateTime.ParseExact(tokens[0], format,
CultureInfo.InvariantCulture);
        var stopDate = DateTime.ParseExact(tokens[1], format,
CultureInfo.InvariantCulture);

        var beginTime = startDate.TimeOfDay;
        var endTime = stopDate.TimeOfDay;

        if (endTime < beginTime) {
            stopDate = stopDate.AddDays(1);
        }

        return Range.Of(startDate, stopDate);
    }
}
```

edited Dec 2 '16 at 1:53

answered Dec 2 '16 at 1:45

Nkosi

120k 17 142 206

0

I would recommend adding a method to your class that represents the actual different between the start and end. For example call it timespan TimeDiff. you will need to include an if statement to confirm that TimeLater is less than TimeEnd and TimeEarlier is greater than TimeStart. Then the 'TimeDiff = TimeLater - TimeEarlier' .

TimeLater is the end of the range provided. And TimeEarlier is the beginning of a range provided.

If you want to calculate the timespan that goes past TimeEnd to TimeStart you will just perform need to perform a check that TimeEarlier is greater than TimeLater and have logic to calculate the difference. It would be along the lines of TimeDiff = (TimeEnd - TimeEarlier) + (TimeLater - TimeEnd)

To accomplish timespan subtraction use .Subtract() in all timespans

edited Dec 2 '16 at 0:19

answered Dec 2 '16 at 0:08

peterpep
189 3 11

0

You're using the wrong types. `Start` and `End` should be `DateTime`, not `TimeSpan`. `End.Subtract(Start)` will provide a `TimeSpan` as its result. The `TimeSpan` type has properties that will provide total number of hours, minutes, etc.

TimeSpan Properties

answered Dec 2 '16 at 0:07

Xavier J
4,049 1 9 25