'Format date and time

Public Interface IEvaluateDateTimeIntervals

    Function timeIntervals(ByVal starttime As String, ByVal endtime As String) As ArrayList

    Function dateIntervals(ByVal startdate As String, ByVal enddate As String, ByVal selectedDays() As String) As ArrayList

End Interface

Public Class datetimeIntervalConversion

    Implements IEvaluateDateTimeIntervals

    Public Function timeIntervals(ByVal starttime As String, ByVal endtime As String) As ArrayList Implements IEvaluateDateTimeIntervals.timeIntervals

        'Calculate time intervals betwen a Starttime and Endtime range

        Dim storeTimeIntervals As New ArrayList 'Return as an arraylist of datatype Datetime

        Dim intervals As New ArrayList

        'If there are blank time values then ignore the entire process

        If Not starttime = String.Empty And Not endtime = String.Empty Then

            Dim time1 As DateTime 'Start time with DateTime datatype

            Dim time2 As DateTime 'End time with DateTime datatype

            Dim t1 As TimeSpan

            Dim t2 As TimeSpan

            'Convert time values to datetime datatype and store the value in 't1 and t2

            time1 = DateTime.ParseExact(starttime, "h:mm tt", Globalization.DateTimeFormatInfo.InvariantInfo)

            time2 = DateTime.ParseExact(endtime, "h:mm tt", Globalization.DateTimeFormatInfo.InvariantInfo)

            t2 = time2.TimeOfDay

            storeTimeIntervals.Add(time1.ToString("h:mm tt")) 'Initialize index 0 with the start time

            Dim a As Integer = 0  'Initialize the counter to 0

            Dim value As DateTime 'Temporary place holder

            Do

                value = Convert.ToDateTime(storeTimeIntervals(a)) 'Place current interval into a temporary variable called 'value'

                value = value.AddMinutes(30)   'add 30 minutes to temprary placeholder called 'value'

                a = a + 1 'increment the index of the array by 1

                storeTimeIntervals.Add(value.ToString("h:mm tt"))  'place new value into the arraylist

                t1 = value.TimeOfDay

            Loop While (t1 < t2) 'Loop until time interval equals End time

        End If

        Return storeTimeIntervals 'Return all interval values as DateTime datatype

    End Function

    Public Function dateIntervals(ByVal startdate As String, ByVal enddate As String, ByVal selectedDays() As String) As ArrayList Implements IEvaluateDateTimeIntervals.dateIntervals

        Dim storeDateIntervals As New ArrayList

        Dim datematch As New ArrayList

        'If there are blank time values then ignore the entire process

        If Not startdate = String.Empty And Not enddate = String.Empty Then

            Dim d1 As Date

            Dim d2 As Date

            Dim dys As Integer = 0 'initialize the first index to 0

            Dim numofdays As Integer

            d1 = Convert.ToDateTime(startdate.Trim).ToShortDateString

            d2 = Convert.ToDateTime(enddate.Trim).ToShortDateString

            Dim day As String = String.Empty

            numofdays = DateDiff("d", d1, d2) 'Get the total number of days in the interval by subtracting the start date from the end date

            '

            storeDateIntervals.Add(d1) 'And start date to the arraylist

            Do While storeDateIntervals(dys) <= d2 'If current date interval is less than the end date the loop

                'Add A Select Statement to filter affirmed days.Leaving out excluded days

                day = WeekdayName(Weekday(storeDateIntervals(dys))) 'Get the day of the week

                Select Case day

                    Case selectedDays(0) 'Sunday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(1) 'Monday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(2) 'Tuesday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(3) 'Wednesday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(4) 'Thursday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(5)  'Friday

                        datematch.Add(storeDateIntervals(dys))

                    Case selectedDays(6) 'Saturday

                        datematch.Add(storeDateIntervals(dys))

                    Case Else

                End Select

                dys = dys + 1 'increment the index by 1

                storeDateIntervals.Add(DateAdd("w", dys, d1))

            Loop

        Else

        End If

        Return datematch

    End Function

End Class