English-Friendly Interval Calculation

Posted At : February 24, 2011 2:23 PM | Posted By : Steve
Related Categories: ColdFusion

I am working on a small program to digest and store RSS feeds (in part using **Ray Camden**'s nice **RSS.cfc**). One thing I ran into is that I want to be able to specify how frequently feeds should get updated. It seems obvious that I should have an argument for this in the init() method of my CFC. What I don't want, however, is to either limit myself to one interval (days, for example) or to have multiple arguments just for the interval.

Moreover, I really want to be able to specify the interval in a human-readable format. So, I created a UDF (implemented as a method in my CFC) to make date calculations based on a readable string indicating the interval.

Here are some examples of intervals that it will calculate:

- daily
- annually
- every day
- every other minute
- every 3rd quarter
- every fourth year
- every other Saturday
- 4th Saturday
- three weeks
- 2 months, 2 days, 11 minutes
- - every second Sunday

Most of these should be pretty obvious, but the last two might require a touch of explanation. The UDF will accept a comma delimited list, in which case it will apply all of the intervals to the date given.

If the interval starts with a "-" then it will subtract the interval given. The subtraction will only apply to the entire interval, not to single items in the list.

The UDF is called "DateAddInterval". Here is the code:

```
<cffunction name="DateAddInterval" returntype="date" output="no">
   <cfargument name="interval" type="string" required="true">
   <cfargument name="date" type="date" default="#now()#">

   <cfset var result = arguments.date>
   <cfset var timespans = "millisecond,second,minute,hour,day,week,month,quarter,year">
   <cfset var dateparts = "l,s,n,h,d,ww,m,q,yyyy">
   <cfset var num = 1>
   <cfset var timespan = "">
   <cfset var datepart = "">
   <cfset var ordinals = "first,second,third,fourth,fifth,sixth,seventh,eighth,ninth,tenth,eleventh,twelfth">
   <cfset var ordinal = "">
   <cfset var numbers = "one,two,three,four,five,six,seven,eight,nine,ten,eleven,twelve">
   <cfset var number = "">
   <cfset var weekdays = "Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday">
   <cfset var weekday = "">
   <cfset var thisint = "">
   <cfset var sNums = 0>
   <cfset var isSubtraction = Left(Trim(arguments.interval),1) EQ "-">
   <cfset var sub = "">

   <cfif isSubtraction>
```

```
      <cfset arguments.interval = Trim(ReplaceNoCase(arguments.interval,"-","","ALL"))>
      <cfset sub = "-">
   </cfif>

   <cfif ListLen(arguments.interval) GT 1>
      <cfloop list="#arguments.interval#" index="thisint">
         <cfset result = DateAddInterval("#sub##thisint#",result)>
      </cfloop>
   <cfelse>
      <cfset arguments.interval = ReplaceNoCase(arguments.interval,"annually","yearly","ALL")>
      <cfset arguments.interval = ReReplaceNoCase(arguments.interval,"\b(\d+)(nd|rd|th)\b","\1","ALL")>
      <cfset sNums = ReFindNoCase("\b\d+\b",arguments.interval,1,true)>
      <!--- Figure out number --->
      <cfif ArrayLen(sNums.pos) AND sNums.pos[1] GT 0>
         <cfset num = Mid(arguments.interval,sNums.pos[1],sNums.len[1])>
      </cfif>
      <cfif ListFindNoCase(arguments.interval,"every"," ")>
         <cfset arguments.interval = ListDeleteAt(arguments.interval,ListFindNoCase(arguments.interval,"every"," ),"
")>
      </cfif>
      <cfloop list="#ordinals#" index="ordinal">
         <cfif ListFindNoCase(arguments.interval,ordinal," ")>
            <cfset num = num * ListFindNoCase(ordinals,ordinal)>
         </cfif>
      </cfloop>
      <cfloop list="#numbers#" index="number">
         <cfif ListFindNoCase(arguments.interval,number," ")>
            <cfset num = num * ListFindNoCase(numbers,number)>
         </cfif>
      </cfloop>
      <cfif ListFindNoCase(arguments.interval,"other"," ")>
         <cfset arguments.interval = ListDeleteAt(arguments.interval,ListFindNoCase(arguments.interval,"other"," "),"
")>
         <cfset num = num * 2>
      </cfif>
      <!--- Check if day of week is specified --->
      <cfloop list="#weekdays#" index="weekday">
         <cfif ListFindNoCase(arguments.interval,weekday," ")>
            <!--- Make sure the date given is on the day of week specified (subtract days as needed) --->
            <cfset arguments.date = DateAdd("d",- Abs( 7 - ListFindNoCase(weekdays,weekday) + DayOfWeek(arguments.date)
) MOD 7,arguments.date)>
            <cfset arguments.interval = ListDeleteAt(arguments.interval,ListFindNoCase(arguments.interval,weekday,"
")," ")>
            <!--- Make sure we are adding weeks --->
            <cfset arguments.interval = ListAppend(arguments.interval,"week"," ")>
         </cfif>
      </cfloop>

      <!--- Figure out timespan --->
      <cfset timespan = ListLast(arguments.interval," ")>

      <!--- Ditch ending "s" or "ly" --->
      <cfif Right(timespan,1) EQ "s">
         <cfset timespan = Left(timespan,Len(timespan)-1)>
      </cfif>
      <cfif Right(timespan,2) EQ "ly">
         <cfset timespan = Left(timespan,Len(timespan)-2)>
      </cfif>
      <cfif timespan EQ "dai">
         <cfset timespan = "day">
      </cfif>

      <cfif ListFindNoCase(timespans,timespan)>
         <cfset datepart = ListGetAt(dateparts,ListFindNoCase(timespans,timespan))>
      <cfelse>
         <cfthrow message="#timespan# is not a valid inteval measurement.">
      </cfif>

      <cfset result = DateAdd(datepart,"#sub##num#",arguments.date)>
   </cfif>

   <cfreturn result>
</cffunction>
```

With the exception of the day of the week piece, all of these calculations are based on regular intervals of time. It will not, for example, calculate the day or week of the month nor of the year. So, "every second Sunday of the month" would not work.

Most of this UDF is pretty simple string parsing to determine the interval. It does, however have two parts that might be of some interest.

The function takes advantage of recursion by calling itself when passed a list. I am a big fan of recursion, but generally care must be taken to prevent infinite loops. In this case, the check of a ListLen() takes care of that since the internal call won't pass a list.

```
<cfif ListLen(arguments.interval) GT 1>
   <cfloop list="#arguments.interval#" index="thisint">
      <cfset result = DateAddInterval("#sub##thisint#",result)>
   </cfloop>
<cfelse>
   ...
</cfif>
```

The other interesting thing is a simple date calculation:

```
<!--- Make sure the date given is on the day of week specified (subtract days as needed) --->
<cfset arguments.date = DateAdd("d",- Abs( 7 - ListFindNoCase(weekdays,weekday) + DayOfWeek(arguments.date) ) MOD
7,arguments.date)>
```

This is a single line of code to determine get the current/previous occurrence of the given day of the week. Here this is in an isolated UDF:

```
function getPriorWeekday(date,weekdaynum) {
   return DateAdd("d",- Abs( 7 - weekdaynum + DayOfWeek(date) ) MOD 7,date);
}
```

If you want to avoid conditionals for this sort of thing then Abs() and MOD are your friends. Abs() turns any number positive and MOD returns the remainder from a division. This is useful for any sort of cyclical calculation - day of the week, minutes on the clock, that sort of thing.

The same thing to get current/next occurrence of a given day of the week would be:

```
function getNextWeekday(date,weekdaynum) {
   return DateAdd("d",( weekdaynum - 1 + DayOfWeek(date) ) MOD 7, date);
}
```

All of this code should be considered open source and free for any use. I'll try to submit it to **CFLib** shortly.