XSLT DateTime Formatting

Finnbarr P. Murphy
(fpm@fpmurphy.com)

Support for date and time formating in the XSLT 1.0 specification is non-existent. This did not mean that a person cannot format date and time strings using XSLT 1.0; it just makes it much harder to do so and adds many extra lines of code to stylesheets. However it is something that everybody who develops stylesheets ends up having to do. In this post I show you several ways to format dates in XSLT 1.0 and discuss some of the new dateTime formating and manipulation functions in XSLT 2.0 and XPath 2.0.

For our first example, suppose we have the following trivial XML document:

```
<?xml version="1.0"?>
<timesheet>
     <name>Ciara</name>
     <date>02081981</date>
</timesheet>
```

and we want to transform the document so that the *date* element is formatted as *MM/DD/YYYY*.

Here is one way of doing it using a named template called *formatdate*.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"  indent="yes"/>
  <xsl:template match="/">
      <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="date">
     <xsl:element name="date">
         <xsl:call-template name="formatdate">
             <xsl:with-param name="datestr" select="."/>
        </xsl:call-template>
     </xsl:element>
  </xsl:template>
  <xsl:template name="formatdate">
     <xsl:param name="datestr" />
     <!-- input format mmddyyyy -->
     <!-- output format mm/dd/yyyy -->
    <xsl:variable name="mm">
       <xsl:value-of select="substring($datestr,1,2)" />
    </xsl:variable>
    <xsl:variable name="dd">
       <xsl:value-of select="substring($datestr,3,2)" />
    </xsl:variable>
    <xsl:variable name="yyyy">
       <xsl:value-of select="substring($datestr,5,4)" />
    </xsl:variable>
    <xsl:value-of select="$mm" />
    <xsl:value-of select="'/'" />
    <xsl:value-of select="$dd" />
    <xsl:value-of select="'/'" />
    <xsl:value-of select="$yyyy" />
  </xsl:template>
  <xsl:template match="node()|@*">
```



```
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Here is the output of the transformation using the *xsltproc* processor.

```
$ xsltproc example1.xsl example1.xml
<?xml version="1.0"?>
<timesheet>
      <name>Ciara</name>
      <date>02/08/1981</date>
</timesheet>
$
```

The next example transforms a XSD dateTime data type to a date string formatted as *MM/DD/YYYY*. A *dateTime* data type is used to specify a date and a time in the following form *YYYY-MM-DDThh:mm:ss* where:

- YYYY indicates the year
- MM indicates the month
- DD indicates the day
- T indicates the start of the required time section
- hh indicates the hour
- mm indicates the minute
- ss indicates the second

This datatype describes instances of ISO 8601 calendar datetimes.

Here is a stylesheet which transforms the date element into a date string formatted as *MM/DD/YYYY*.

```
<?xml version="1.0"?>
<articles>
    <article>
        <id>1482</id>
        <date>2008-03-29T00:00:00+01:00</date>
    </article>
    <article>
        <id>1483</id>
        <date>2008-03-30T00:00:00+01:00</date>
    </article>
</articles>
```

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="*">
      <xsl:copy><xsl:apply-templates /></xsl:copy>
  </xsl:template>
  <xsl:template match="date">
     <xsl:copy>
         <xsl:call-template name="formatdate">
             <xsl:with-param name="DateTimeStr" select="."/>
        </xsl:call-template>
```



```
    </xsl:copy>
  </xsl:template>
  <xsl:template name="formatdate">
     <xsl:param name="DateTimeStr" />
     <!-- input format xslt datetime string -->
     <!-- output format mm/dd/yyyy -->
     <xsl:variable name="datestr">
         <xsl:value-of select="substring-before($DateTimeStr,'T')" />
     </xsl:variable>
     <xsl:variable name="mm">
         <xsl:value-of select="substring($datestr,6,2)" />
     </xsl:variable>
     <xsl:variable name="dd">
        <xsl:value-of select="substring($datestr,9,2)" />
     </xsl:variable>
     <xsl:variable name="yyyy">
        <xsl:value-of select="substring($datestr,1,4)" />
     </xsl:variable>
     <xsl:value-of select="concat($mm,'/', $dd, '/', $yyyy)" />
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<articles>
    <article>
        <id>1482</id>
        <date>03/29/2008</date>
    </article>
    <article>
        <id>1483</id>
        <date>03/30/2008</date>
    </article>
</articles>
```

With XSLT 2.0 and XPATH 2,0, things got much easier as support for date and time formatting and manipulation was finally incorporated into the specifications. Here is the first part of section 16.5 of the XSLT 2.0 specification:

16.5 Formatting Dates and Times

Three functions are provided to represent dates and times as a string, using the conventions of a selected calendar, language, and country. Each has two variants.

```
format-dateTime($value    as xs:dateTime?,
                $picture  as xs:string,
                $language as xs:string?,
                $calendar as xs:string?,
                $country  as xs:string?) as xs:string?
format-dateTime($value  as xs:dateTime?, $picture as xs:string) as xs:string?
format-date($value    as xs:date?,
            $picture  as xs:string,
            $language as xs:string?,
            $calendar as xs:string?,
            $country  as xs:string?) as xs:string?
format-date($value as xs:date?, $picture as xs:string) as xs:string?
format-time($value    as xs:time?,
            $picture  as xs:string,
            $language as xs:string?,
            $calendar as xs:string?,
            $country  as xs:string?) as xs:string?
format-time($value as xs:time?, $picture as xs:string) as xs:string?
```

The format-dateTime, format-date, and format-time functions format $value as a string using the picture string specified by the $picture argument, the calendar specified by the $calendar argument, the language specified by the $language argument, and the country specified by the $country argument. The result of the function is the formatted string representation of the supplied dateTime, date, or time value.

[Definition: The three functions format-date, format-time, and format-dateTime are referred to collectively as the **date formatting functions**.]

If $value is the empty sequence, the empty sequence is returned.

Calling the two-argument form of each of the three functions is equivalent to calling the five-argument form with each of the last three arguments set to an empty sequence.

Here is an XSLT 2.0 stylesheet which uses the two-argument form of *format-dateTime* to transform the XML document used in the previous example.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="*">
        <xsl:copy><xsl:apply-templates /></xsl:copy>
  </xsl:template>
  <xsl:template match="date">
       <xsl:copy>
           <xsl:value-of select="format-dateTime(.,'[M01]/[D01]/[Y0001]')" />
       </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

The output is the same as before but note the decrease in the number of lines of code required to produce the output.

The second argument to *format-dateTime* is known as the picture string argument. A picture string is a string that identifies the components to be outputted and the format in which they are to be outputted. It consists of a sequence of variable markers and literal substrings. See section 16.5.1 of the XSLT 2.0 specification for all the gory details. There is support for more than 20 calendars including the Christian Era (AD – Anno Domini) and the Muhammedan Era (AH – Anno Hegirae).

You can even handle timezone conversions using the XPath 2.0 *adjust-dateTime-to-timezone* function. Note, however, that this function does not handle daylight saving time.

Here is a simple stylesheet which takes the current time and outputs an XML document giving the corresponding time in Dublin, Boston, Stanford and Mumbai. This is a trivial example but it shows the power of the *adjust-dateTime-to-timezone* function.

```
<xsl:stylesheet
  version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="#all" >
  <xsl:output indent="yes"/>
  <xsl:variable name="IST-offset" select="'PT5H30M'"/>
  <xsl:variable name="BST-offset" select="'PT1H'"/>
  <xsl:variable name="EST-offset" select="'-PT4H'"/>
  <xsl:variable name="PST-offset" select="'-PT7H'"/>
  <xsl:template match="/">
      <xsl:variable name="current" select="current-dateTime()"/>
      <xsl:call-template name="timezones">
          <xsl:with-param name="UTC-timestamp" select="$current" />
      </xsl:call-template>
  </xsl:template>
  <xsl:template name="timezones">
      <xsl:param name="UTC-timestamp" />
      <xsl:variable name="IST-timestamp" select="adjust-dateTime-to-timezone($UTC-timestam
p, xs:dayTimeDuration($IST-offset))"/>
      <xsl:variable name="BST-timestamp" select="adjust-dateTime-to-timezone($UTC-timestam
p, xs:dayTimeDuration($BST-offset))"/>
      <xsl:variable name="EST-timestamp" select="adjust-dateTime-to-timezone($UTC-timestam
p, xs:dayTimeDuration($EST-offset))"/>
      <xsl:variable name="PST-timestamp" select="adjust-dateTime-to-timezone($UTC-timestam
p, xs:dayTimeDuration($PST-offset))"/>
      <meeting UTCtime="{$UTC-timestamp}" >
```



```
          <xsl:variable name="ist-offset" select="format-dateTime($IST-timestamp, '[ZN,*-3
]', (), (), 'uk')"/>
          <datetime location="Mubai" timezone="IST" UTCoffset="{$ist-offset}">
              <xsl:value-of select="format-dateTime($IST-timestamp, '[D] [MN,*-3] [Y] [h]:
[m01][PN,*-2]', (), (), 'uk')"/>
          </datetime>
          <xsl:variable name="bst-offset" select="format-dateTime($BST-timestamp, '[ZN,*-3
]', (), (), 'uk')"/>
          <datetime location="Dublin" timezone="BST" UTCoffset="{$bst-offset}">
              <xsl:value-of select="format-dateTime($BST-timestamp, '[D] [MN,*-3] [Y] [h]:
[m01][PN,*-2]', (), (), 'uk')"/>
          </datetime>
          <xsl:variable name="est-offset" select="format-dateTime($EST-timestamp, '[ZN,*-3
]', (), (), 'us')"/>
          <datetime location="Boston" timezone="EST" UTCoffset="{$est-offset}">
              <xsl:value-of select="format-dateTime($EST-timestamp, '[D] [MNn] [Y] [h]:[m0
1][PN,*-2]', (), (), 'us')"/>
          </datetime>
          <xsl:variable name="pst-offset" select="format-dateTime($PST-timestamp, '[ZN,*-3
]', (), (), 'us')"/>
          <datetime location="Stanford" timezone="PST" UTCoffset="{$pst-offset}">
              <xsl:value-of select="format-dateTime($PST-timestamp, '[D] [MNn] [Y] [h]:[m0
1][PN,*-2]', (), (), 'us')"/>
          </datetime>
      </meeting>
  </xsl:template>
</xsl:stylesheet>
```

Here is the output from this transformation:

```
<?xml version="1.0" encoding="UTF-8"?>
<meeting UTCtime="2008-05-04T19:50:50.663-04:00">
   <datetime location="Mubai" timezone="IST" UTCoffset="+05:30">5 MAY 2008 5:20AM</datetim
e>
   <datetime location="Dublin" timezone="BST" UTCoffset="+01:00">5 MAY 2008 12:50AM</datet
ime>
   <datetime location="Boston" timezone="EST" UTCoffset="-04:00">4 May 2008 7:50PM</dateti
me>
   <datetime location="Stanford" timezone="PST" UTCoffset="-07:00">4 May 2008 4:50PM</date
time>
</meeting>
```

In case you are unaware of it, India Standard Time (IST) is 5 hrs 30 minutes ahead of UTC and there is no daylight savings time for 2008.

You may be wondering why I have not discussed the EXSLT date and time extensions before now. The reason is simple. While EXSLT specifies *date:format-date* and *date:date-format* functions, no currently available XSLT processor natively supports these two functions. In fact, there is no implementation of *date:date-format*. With the increasing popularity of XSLT 2.0 and XPath 2.0 there simply is no reason to consider these functions any further.

If you want to learn more about the XSLT 2.0 and XPath 2.0 date and time formating functions, I recommend that you read XSLT 2.0 Programmers Reference by Michael Kay.

