XSL Library Template Reference

COLLABORATORS

	TITLE : XSL Library Template Reference	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 General Library Templates **1**

1.1 dot.count . . . 1
1.2 copy-string . . . 1
1.3 string.subst . . . 2
1.4 xpointer.idref . . . 3
1.5 length-magnitude . . . 3
1.6 length-units . . . 4
1.7 length-spec . . . 4
1.8 length-in-points . . . 5
1.9 pi-attribute . . . 6
1.10 lookup.key . . . 7
1.11 xpath.location . . . 8
1.12 comment-escape-string . . . 8
1.13 comment-escape-string.recursive . . . 9

2 Relative URI Functions **10**

2.1 count.uri.path.depth . . . 10
2.2 trim.common.uri.paths . . . 11

A The Stylesheet **12**

Introduction

This is technical reference documentation for the DocBook XSL Stylesheets; it documents (some of) the parameters, templates, and other elements of the stylesheets.

This is not intended to be “user” documentation. It is provided for developers writing customization layers for the stylesheets, and for anyone who’s interested in “how it works”.

Although I am trying to be thorough, this documentation is known to be incomplete. Don’t forget to read the source, too :-)

Chapter 1

General Library Templates

1.1 dot.count

dot.count — Returns the number of "." characters in a string

Description

```
<xsl:template name="dot.count">
  <!-- Returns the number of "." characters in a string -->
  <xsl:param name="string"/>
  <xsl:param name="count"
             select="0"/>
  <xsl:choose>
    <xsl:when test="contains($string, '.')">
      <xsl:call-template name="dot.count">
        <xsl:with-param name="string"
                        select="substring-after($string, '.')"/>
        <xsl:with-param name="count"
                        select="$count+1"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$count"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.2 copy-string

copy-string — Returns "count" copies of a string

Description

```
<xsl:template name="copy-string">
  <!-- returns 'count' copies of 'string' -->
  <xsl:param name="string"/>
  <xsl:param name="count"
             select="0"/>
```

```
  <xsl:param name="result"/>

  <xsl:choose>
    <xsl:when test="$count>0">
      <xsl:call-template name="copy-string">
        <xsl:with-param name="string"
                        select="$string"/>
        <xsl:with-param name="count"
                        select="$count - 1"/>
        <xsl:with-param name="result">
          <xsl:value-of select="$result"/>
          <xsl:value-of select="$string"/>
        </xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$result"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.3 string.subst

string.subst — Substitute one text string for another in a string

Description

The `string.subst` template replaces all occurances of *`target`* in *`string`* with *`replacement`* and returns the result.

```
<xsl:template name="string.subst">
  <xsl:param name="string"/>
  <xsl:param name="target"/>
  <xsl:param name="replacement"/>

  <xsl:choose>
    <xsl:when test="contains($string, $target)">
      <xsl:variable name="rest">
        <xsl:call-template name="string.subst">
          <xsl:with-param name="string"
                          select="substring-after($string, $target)"/>
          <xsl:with-param name="target"
                          select="$target"/>
          <xsl:with-param name="replacement"
                          select="$replacement"/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:value-of select="concat(substring-before($string, $target),  ↩
                                              $replacement,  ↩
                                              $rest)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$string"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.4 xpointer.idref

xpointer.idref — Extract IDREF from an XPointer

Description

The `xpointer.idref` template returns the ID portion of an XPointer which is a pointer to an ID within the current document, or the empty string if it is not.

In other words, `xpointer.idref` returns "foo" when passed either `#foo` or `#xpointer(id('foo'))`, otherwise it returns the empty string.

```
<xsl:template name="xpointer.idref">
  <xsl:param name="xpointer">http://...</xsl:param>
  <xsl:choose>
    <xsl:when test="starts-with($xpointer, '#xpointer(id(')">
      <xsl:variable name="rest"
                    select="substring-after($xpointer, '#xpointer(id(')"/>
      <xsl:variable name="quote"
                    select="substring($rest, 1, 1)"/>
      <xsl:value-of select="substring-before(substring-after($xpointer, $quote), $quote)"/>
    </xsl:when>
    <xsl:when test="starts-with($xpointer, '#')">
      <xsl:value-of select="substring-after($xpointer, '#')"/>
    </xsl:when>
    <!-- otherwise it's a pointer to some other document -->
  </xsl:choose>
</xsl:template>
```

1.5 length-magnitude

length-magnitude — Return the unqualified dimension from a length specification

Description

The `length-magnitude` template returns the unqualified length ("20" for "20pt") from a dimension.

```
<xsl:template name="length-magnitude">
  <xsl:param name="length"
             select="'0pt'"/>

  <xsl:choose>
    <xsl:when test="string-length($length) = 0"/>
    <xsl:when test="substring($length,1,1) = '0'                    or substring($length ↩
        ,1,1) = '1'                    or substring($length,1,1) = '2'                    ↩
        or substring($length,1,1) = '3'                    or substring($length,1,1) = '4'  ↩
                        or substring($length,1,1) = '5'                    or substring ↩
        ($length,1,1) = '6'                    or substring($length,1,1) = '7'  ↩
                        or substring($length,1,1) = '8'                    or substring ↩
        ($length,1,1) = '9'                    or substring($length,1,1) = '.'">
      <xsl:value-of select="substring($length,1,1)"/>
      <xsl:call-template name="length-magnitude">
        <xsl:with-param name="length"
                        select="substring($length,2)"/>
      </xsl:call-template>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```

1.6 length-units

length-units — Return the units from a length specification

Description

The `length-units` template returns the units ("pt" for "20pt") from a length. If no units are supplied on the length, the *`defaiult.units`* are returned.

```
<xsl:template name="length-units">
  <xsl:param name="length"
             select="'0pt'"/>
  <xsl:param name="default.units"
             select="'px'"/>
  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length"
                      select="$length"/>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="units">
    <xsl:value-of select="substring($length, string-length($magnitude)+1)"/>
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$units = ''">
      <xsl:value-of select="$default.units"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$units"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.7 length-spec

length-spec — Return a fully qualified length specification

Description

The `length-spec` template returns the qualified length from a dimension. If an unqualified length is given, the *`default.units`* will be added to it.

```
<xsl:template name="length-spec">
  <xsl:param name="length"
             select="'0pt'"/>
  <xsl:param name="default.units"
             select="'px'"/>

  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length"
                      select="$length"/>
    </xsl:call-template>
  </xsl:variable>
```

```
  <xsl:variable name="units">
    <xsl:value-of select="substring($length, string-length($magnitude)+1)"/>
  </xsl:variable>

  <xsl:value-of select="$magnitude"/>
  <xsl:choose>
    <xsl:when test="$units='cm'                    or $units='mm'                    or  ↩
        $units='in'                    or $units='pt'                    or $units='pc'  ↩
                          or $units='px'                    or $units='em'">
      <xsl:value-of select="$units"/>
    </xsl:when>
    <xsl:when test="$units = ''">
      <xsl:value-of select="$default.units"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
        <xsl:text>Unrecognized unit of measure: </xsl:text>
        <xsl:value-of select="$units"/>
        <xsl:text>.</xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.8 length-in-points

length-in-points — Returns the size, in points, of a specified length

Description

The `length-in-points` template converts a length specification to points and returns that value as an unqualified number.

Caution
There is no way for the template to infer the size of an `em`. It relies on the default *`em.size`* which is initially `10` (for 10pt).
Similarly, converting pixesl to points relies on the *`pixels.per.inch`* parameter which is initially `90`.

```
<xsl:template name="length-in-points">
  <xsl:param name="length"
             select="'0pt'"/>
  <xsl:param name="em.size"
             select="10"/>
  <xsl:param name="pixels.per.inch"
             select="90"/>

  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length"
                      select="$length"/>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="units">
    <xsl:value-of select="substring($length, string-length($magnitude)+1)"/>
```

```
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$units = 'pt'">
      <xsl:value-of select="$magnitude"/>
    </xsl:when>
    <xsl:when test="$units = 'cm'">
      <xsl:value-of select="$magnitude div 2.54 * 72.0"/>
    </xsl:when>
    <xsl:when test="$units = 'mm'">
      <xsl:value-of select="$magnitude div 25.4 * 72.0"/>
    </xsl:when>
    <xsl:when test="$units = 'in'">
      <xsl:value-of select="$magnitude * 72.0"/>
    </xsl:when>
    <xsl:when test="$units = 'pc'">
      <xsl:value-of select="$magnitude div 6.0 * 72.0"/>
    </xsl:when>
    <xsl:when test="$units = 'px'">
      <xsl:value-of select="$magnitude div $pixels.per.inch * 72.0"/>
    </xsl:when>
    <xsl:when test="$units = 'em'">
      <xsl:value-of select="$magnitude * $em.size"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
        <xsl:text>Unrecognized unit of measure: </xsl:text>
        <xsl:value-of select="$units"/>
        <xsl:text>.</xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.9 pi-attribute

pi-attribute — Extract a pseudo-attribute from a PI

Description

The `pi-attribute` template extracts a pseudo-attribute from a processing instruction. For example, given the PI "`<?foo bar="1" baz='red'?>`",

```
<xsl:call-template name="pi-attribute">
  <xsl:with-param name="pis" select="processing-instruction('foo')"/>
  <xsl:with-param name="attribute" select="'baz'"/>
</xsl:call-template>
```

will return "red". This template returns the first matching attribute that it finds. Presented with processing instructions that contain badly formed pseudo-attributes (missing or unbalanced quotes, for example), the template may silently return erroneous results.

```
<xsl:template name="pi-attribute">
  <xsl:param name="pis"
             select="processing-instruction('')"/>
  <xsl:param name="attribute">filename</xsl:param>
  <xsl:param name="count">1</xsl:param>
```

```
  <xsl:choose>
    <xsl:when test="$count>count($pis)">
      <!-- not found -->
    </xsl:when>
    <xsl:otherwise>
      <xsl:variable name="pi">
        <xsl:value-of select="$pis[$count]"/>
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="contains($pi,concat($attribute, '='))">
          <xsl:variable name="rest"
                        select="substring-after($pi,concat($attribute,'='))"/>
          <xsl:variable name="quote"
                        select="substring($rest,1,1)"/>
          <xsl:value-of select="substring-before(substring($rest,2),$quote)"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:call-template name="pi-attribute">
            <xsl:with-param name="pis"
                            select="$pis"/>
            <xsl:with-param name="attribute"
                            select="$attribute"/>
            <xsl:with-param name="count"
                            select="$count + 1"/>
          </xsl:call-template>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.10 lookup.key

lookup.key — Retrieve the value associated with a particular key in a table

Description

Given a table of space-delimited key/value pairs, the `lookup.key` template extracts the value associated with a particular key.

```
<xsl:template name="lookup.key">
  <xsl:param name="key"
             select="''"/>
  <xsl:param name="table"
             select="''"/>

  <xsl:if test="contains($table, ' ')">
    <xsl:choose>
      <xsl:when test="substring-before($table, ' ') = $key">
        <xsl:variable name="rest"
                      select="substring-after($table, ' ')"/>
        <xsl:choose>
          <xsl:when test="contains($rest, ' ')">
            <xsl:value-of select="substring-before($rest, ' ')"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="$rest"/>
          </xsl:otherwise>
        </xsl:choose>
```

```
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="lookup.key">
          <xsl:with-param name="key"
                          select="$key"/>
          <xsl:with-param name="table"
                          select="substring-after(substring-after($table,' '), ' ')"/>
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:if>
</xsl:template>
```

1.11 xpath.location

xpath.location — Calculate the XPath child-sequence to the current node

Description

The `xpath.location` template calculates the absolute path from the root of the tree to the current element node.

```
<xsl:template name="xpath.location">
  <xsl:param name="node"
             select="."/>
  <xsl:param name="path"
             select="''"/>

  <xsl:variable name="next.path">
    <xsl:value-of select="local-name($node)"/>
    <xsl:if test="$path != ''">/</xsl:if>
    <xsl:value-of select="$path"/>
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$node/parent::*">
      <xsl:call-template name="xpath.location">
        <xsl:with-param name="node"
                        select="$node/parent::*"/>
        <xsl:with-param name="path"
                        select="$next.path"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>/</xsl:text>
      <xsl:value-of select="$next.path"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

1.12 comment-escape-string

comment-escape-string — Prepare a string for inclusion in an XML comment

Description

The `comment-escape-string` template returns a string that has been transformed so that it can safely be output as an XML comment. Internal occurrences of "--" will be replaced with "- -" and a leading and/or trailing space will be added to the string, if necessary.

```
<xsl:template name="comment-escape-string">
  <xsl:param name="string"
             select="''"/>

  <xsl:if test="starts-with($string, '-')">
    <xsl:text> </xsl:text>
  </xsl:if>

  <xsl:call-template name="comment-escape-string.recursive">
    <xsl:with-param name="string"
                    select="$string"/>
  </xsl:call-template>

  <xsl:if test="substring($string, string-length($string), 1) = '-'">
    <xsl:text> </xsl:text>
  </xsl:if>
</xsl:template>
```

1.13 comment-escape-string.recursive

comment-escape-string.recursive — Internal function used by comment-escape-string

Description

The `comment-escape-string.recursive` template is used by `comment-escape-string`.

```
<xsl:template name="comment-escape-string.recursive">
  <xsl:param name="string"
             select="''"/>
  <xsl:choose>
    <xsl:when test="contains($string, '--')">
      <xsl:value-of select="substring-before($string, '--')"/>
      <xsl:value-of select="'- -'"/>
      <xsl:call-template name="comment-escape-string.recursive">
        <xsl:with-param name="string"
                        select="substring-after($string, '--')"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$string"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Chapter 2

Relative URI Functions

These functions manipulate relative URI references.

The following assumptions must hold true:

1. All URIs are relative.
2. No URI contains the "`../`" sequence which would effectively move "up" the hierarchy.

If these assumptions do not hold, the results are unpredictable.

2.1 count.uri.path.depth

count.uri.path.depth — Count the number of path components in a relative URI

Description

This function counts the number of path components in a relative URI.

```
<xsl:template name="count.uri.path.depth">
  <xsl:param name="filename"
             select="''"/>
  <xsl:param name="count"
             select="0"/>

  <xsl:choose>
    <xsl:when test="contains($filename, '/')">
      <xsl:call-template name="count.uri.path.depth">
        <xsl:with-param name="filename"
                        select="substring-after($filename, '/')"/>
        <xsl:with-param name="count"
                        select="$count + 1"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$count"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

2.2 trim.common.uri.paths

trim.common.uri.paths — Trim common leading path components from a relative URI

Description

This function trims common leading path components from a relative URI.

```
<xsl:template name="trim.common.uri.paths">
  <xsl:param name="uriA"
             select="''"/>
  <xsl:param name="uriB"
             select="''"/>
  <xsl:param name="return"
             select="'A'"/>

  <xsl:choose>
    <xsl:when test="contains($uriA, '//') and contains($uriB, '//')                    and ↩
        substring-before($uriA, '//') = substring-before($uriB, '//')">
      <xsl:call-template name="trim.common.uri.paths">
        <xsl:with-param name="uriA"
                        select="substring-after($uriA, '//')"/>
        <xsl:with-param name="uriB"
                        select="substring-after($uriB, '//')"/>
        <xsl:with-param name="return"
                        select="$return"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test="$return = 'A'">
          <xsl:value-of select="$uriA"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$uriB"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Appendix A

The Stylesheet

The `lib.xsl` stylesheet is just a wrapper around these functions. <!-- ** $Id: lib.xml,v 1.1.2.2 2002/09/04 13:51:26 jdj Exp $ ** This file is part of the XSL DocBook Stylesheet distribution. See ../README or http://nwalsh.com/docbook/xsl/ for copyright and other information. This module implements DTD-independent functions ** --> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-prefixes="src" version="1.0"> </xsl:stylesheet>