KEYS SORTING, GROUPING, NUMBERING, AND MULTIPLE DOCUMENTS PROCESSING

XSLT

INTRODUCTION

introduction

The xsl:sort XSLT command enables you to sort a collection of related items. This element's attributes allow you to specify specifics regarding the sorting process, such as whether you want the results to be sorted alphabetically or numerically, using multiple keys, or in reverse.

SAMPLE CODE XML

```
<employees>
  <employee hireDate="04/23/1999">
    <last>Hill</last>
    <first>Phil</first>
    <salary>100000</salary>
  </employee>

  <employee hireDate="09/01/1998">
    <last>Herbert</last>
    <first>Johnny</first>
    <salary>95000</salary>
  </employee>

  <employee hireDate="08/20/2000">
    <last>Hill</last>
    <first>Graham</first>
    <salary>89000</salary>
  </employee>
</employees>
```

SAMPLE CODE XSL

```
<!-- xq424.xsl: converts xq423.xml into xq425.xml -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
 <xsl:output method="text"/>

 <xsl:template match="employees">
  <xsl:apply-templates>
   <xsl:sort select="salary"/>
  </xsl:apply-templates>
 </xsl:template>

 <xsl:template match="employee">
  Last:      <xsl:apply-templates select="last"/>
  First:     <xsl:apply-templates select="first"/>
  Salary:    <xsl:apply-templates select="salary"/>
  Hire Date: <xsl:apply-templates select="@hireDate"/>
  <xsl:text>
 </xsl:text>
 </xsl:template>
</xsl:stylesheet>
```

It's fairly easy. The XSLT processor is instructed to sort the child elements of the employees element by the xsl:apply-templates instruction and the xsl:sort child in the template for the workers element. The pay values of the employee components are the sort key, as specified by the select attribute of the xsl:sort command. (If the choose attribute is not present, the XSLT processor sorts the elements using a string representation of the sort key.) A final xsl:text element adds a carriage return after each hire date value, and the template rule for the employee element adds each of its child node's values to the result tree before a label.

OUTPUT

Last: Hill
First: Phil
Salary: 100000
Hire Date: 04/23/1999

Last: Hill
First: Graham
Salary: 89000
Hire Date: 08/20/2000

Last: Herbert
First: Johnny
Salary: 95000
Hire Date: 09/01/1998

sorting in xslt

The employees are sorted by salary, but they're sorted alphabetically -- "1" comes before "8" and "9", so a salary of "100000" comes first. But we want the salary values treated as numbers, so we make a simple addition to the template's xsl:sort instruction:

SORTING NUMBERS

```
<!-- xq426.xsl: converts xq423.xml into xq427.xml -->

<xsl:template match="employees">
  <xsl:apply-templates>
    <xsl:sort select="salary" data-type="number"/>
  </xsl:apply-templates>
</xsl:template>
```

OUTPUT

OUTPUT

Now, the output is sorted by the salary element's numeric value:

Last: Hill
First: Graham
Salary: 89000
Hire Date: 08/20/2000

Last: Herbert
First: Johnny
Salary: 95000
Hire Date: 09/01/1998

Last: Hill
First: Phil
Salary: 100000
Hire Date: 04/23/1999

ORDER OF SORTING

To reverse the order of this or any other sort, add an order attribute with a value of "descending":

```
<!-- xq428.xsl: converts xq423.xml into xq429.xml -->

<xsl:template match="employees">
  <xsl:apply-templates>
    <xsl:sort select="salary" data-type="number" order="descending"/>
  </xsl:apply-templates>
</xsl:template>
```

OUTPUT

OUTPUT

Whether the data-type attribute has a value of "number" like the stylesheet above or "text" (the default), an order value of "descending" reverses the order of the sort:

Last: Hill
First: Phil
Salary: 100000
Hire Date: 04/23/1999

Last: Herbert
First: Johnny
Salary: 95000
Hire Date: 09/01/1998

Last: Hill
First: Graham
Salary: 89000
Hire Date: 08/20/2000

GROUPING

If your xsl:apply-templates (or xsl:for-each) element has more than one xsl:sort instruction inside of it, the XSLT processor treats them as multiple keys to the sort. For example, the stylesheet with this next template sorts the employees by last name and then by first name so that any employees with the same last name will be in first name order.

```
<!-- xq430.xsl: converts xq423.xml into xq431.xml -->

<xsl:template match="employees">
  <xsl:apply-templates>
    <xsl:sort select="last"/><xsl:sort select="first"/>
  </xsl:apply-templates>
</xsl:template>
```

OUTPUT

OUTPUT

When applied to the document above, the result shows Johnny Herbert before Phil and Graham Hill, and the secondary sort puts Graham Hill before Phil Hill:

Last: Herbert
First: Johnny
Salary: 95000
Hire Date: 09/01/1998

Last: Hill
First: Graham
Salary: 89000
Hire Date: 08/20/2000

Last: Hill
First: Phil
Salary: 100000
Hire Date: 04/23/1999

KEY SORTING

The sort key doesn't need to be an element child of the sorted elements. The xsl:sort instruction's select attribute can take any XPath expression as a sort key. For example, the following version sorts the employees by their hireDate attribute values:

```
<!-- xq432.xsl: converts xq423.xml into xq433.xml -->


<xsl:template match="employees">
  <xsl:apply-templates>
    <xsl:sort select="@hireDate"/>
  </xsl:apply-templates>
</xsl:template>
```

OUTPUT

OUTPUT

Treating the dates as strings doesn't do much good, because they're sorted alphabetically,

Last: Hill
First: Phil
Salary: 100000
Hire Date: 04/23/1999

Last: Hill
First: Graham
Salary: 89000
Hire Date: 08/20/2000

Last: Herbert
First: Johnny
Salary: 95000
Hire Date: 09/01/1998

THANKS FOR WATCHING

Thank you