Exercises with XSLT

Movies xml

<?xsl-stylesheet ->

Links to xslt

Movies xml

xslt

movies xml link dummy with xslt

xslt uses document(movies.xml)

Different ways

Before we start doing XSLT

Movies xml .xml <?xsl-stylesheet>

XSLT .xsl

Movies xml

Xslt

dummy.xml is one I want to open

links to xslt

document(“movies.xml”)

open dummy one gets xslt one using movies

First one link movies xml to xslt and then look at second one.

Want to have an output that is XML and we want each actor ,every actor and actors are inside the movies.

<Actors>  
<Actor Name=”” YOB=”” Country=”” >

<Actor ……./>

</Actors>

1. **xsl:copy-of**: This element is used to create a copy of the nodes selected by the **select** attribute. The copied nodes are inserted into the output document.
2. **select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]"**: This is an XPath expression used to select which nodes should be copied. It specifies the selection of **Actor** elements under **Movie** elements, which are themselves under **Movies**. The nodes are selected based on a condition:

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:copy-of select="Movies/Movie/Actor[not(@Name=following::Actor/@Name]">

</Actors>

</xsl:template>

</xsl:transform>

Start from the top,start building the content

* **[not(@Name=following::Actor/@Name)]**: This is a predicate (a condition inside square brackets) that filters which **Actor** nodes to select. It uses the **not()** function to include only those **Actor** nodes where there are no following **Actor** nodes with the same **@Name** attribute within the entire document.

**Helps to remove duplicates**

To sort them it requires a loop.

Couple of options when we are doing loops.

Want to do a for-each loop.

Looping through actor element nodes.

Relative to thing in the loop.

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name">

</xsl:for-each>

</Actors>

</xsl:template>

</xsl:transform>

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name"/>

<xsl:copy-of select="current()"/>

</xsl:for-each>

</Actors>

</xsl:template>

</xsl:transform>

Copy-of

Don’t want all the attributes,want to keep only the names.

In that case want to create a new actor.

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name"/>

<Actor Name=”{@Name}”></Actor>

</xsl:for-each>

</Actors>

</xsl:template>

</xsl:transform>

Want to put name not as an attribute but as an actual content,

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name"/>

<Actor Name="{@Name}"></Actor>

</xsl:for-each>

</Actors>

</xsl:template>

</xsl:transform>

For {} curly braces need to have XSLT 3

If only have XSLT 1 need to use some

<Actor> <xsl:value-of select=”@Name”></Actor>

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<Actors>

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name"/>

<xsl:value-of select="@Country"/>

</xsl:for-each>

</Actors>

</xsl:template>

</xsl:transform>

How about not using hard coded elements Actors and Actor.

Maybe don’t want to have actors created this way.

Can use instructions that are node creation instructions.

<?xml version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="xml"/>

<xsl:template match="/">

<xsl:element name="Actors">

<xsl:for-each select="Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">

<xsl:sort select="@Name"/>

<xsl:element name="Actor">

<xsl:attribute name="TheName">

<xsl:value-of select="@Name"></xsl:value-of>

</xsl:attribute>

</xsl:element>

</xsl:for-each>

</xsl:element>

</xsl:template>

</xsl:transform>

XML document with every Actor as an element.

HTML document with all actors as a HTML table.

Actors

One row with headings

Something like Name,YOB and Country.

Name YOB Country

Sylvester Stallone

Woody Allen

2nd way of linking – dummy file

<?xml-version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="html"/>

<xsl:variable name="d" select="document('movies.xml')"/>

<xsl:template match="/">

<html>

<head>

<title>

Actors</title>

</head>

<body>

<h1>Actors</h1>

</body>

</html>

</xsl:template>

</xsl:transform>

Open movies.xml and put it in a variable d.

Then it will start doing content of this template,which is right now hard coded.

It will create Actors,the heading before the table.

Don’t have the table yet in here.

Xt2.xsl

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="xt2.xsl"?>

<Dummy/>

<?xml-version="1.0" encoding="UTF-8"?>

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1">

<xsl:output method="html"/>

<xsl:variable name="d" select="document('movies.xml')"/>

<xsl:template match="/">

<html>

<head>

<title>

Actors</title>

</head>

<body>

<h1>Actors</h1>

<table border="2" width="60%">

<tr>

<th>Name</th>

<th>YoB</th>

<th>Country</th>

</tr>

</body>

</html>

</xsl:template>

</xsl:transform>

<xsl:apply-templates select=” Movies/Movie/Actor[not(@Name=following::Actor/@Name)]">”>

<xsl:sort select=”@Name”>

</xsl:apply-templates>

</table> </body> </html>

</xsl:template></xsl:transform>