Data Mapping – Using Iteration

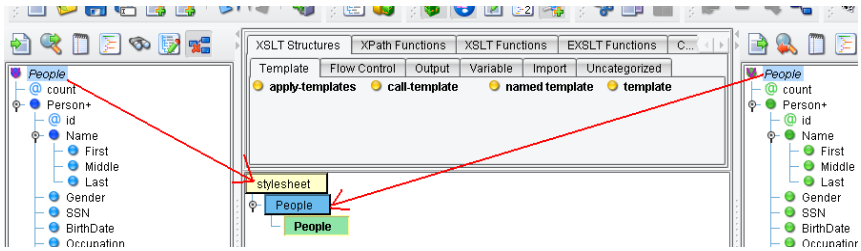
Overview

In this tutorial we'll cover using the "for-each" instruction in XSLT to perform iteration over elements. This tutorial extends on concepts covered in "Data Mapping – Creating a Simple Mapping," so users are expected to be familiar with that material.

Steps

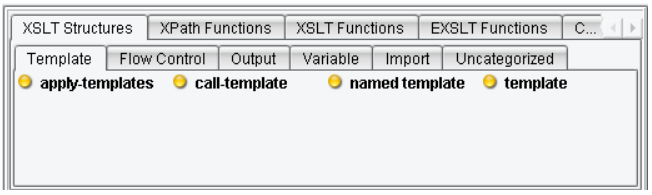
Create a new Route similar to the one used in the "Data Mapping – Creating a Simple Mapping" tutorial, along with a new Format we'll call "PeopleA to PeopleA".

1. Load in "PeopleA.xml" for both the Source and Target formats:

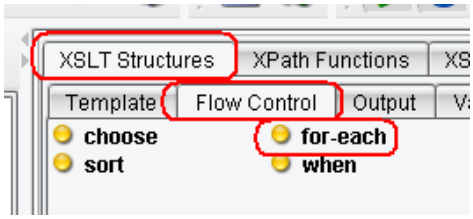


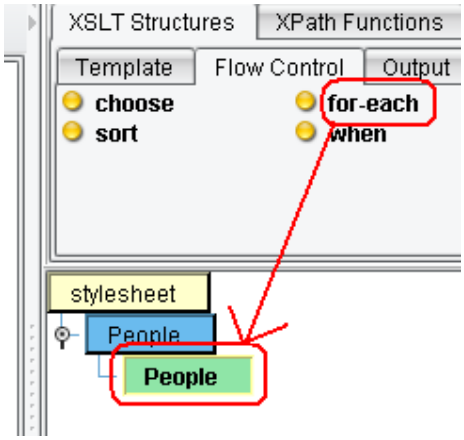
2. Map "People" from the Source onto the "stylesheet" element in the center (mapping) panel, then "People" from the Target onto the "People" now in the center.

   In the previous tutorial, we mapped only the first Person onto the Target. This time, we'll map each Person from the Source to a corresponding Person in the Target. To do this, we'll make use of the Tool Palette at the top of the screen:

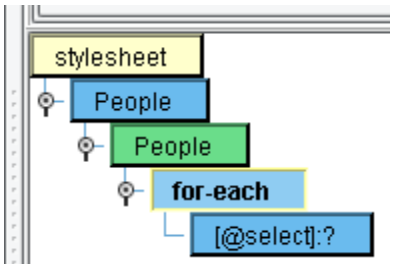


The tool palette is composed of two rows of tabs, which organize XSLT instructions, functions, extensions, call-outs, and other drag-and-drop components, by category. The instruction we're looking for is called “for-each” and it can be found under XSLT Structures → Flow Control:





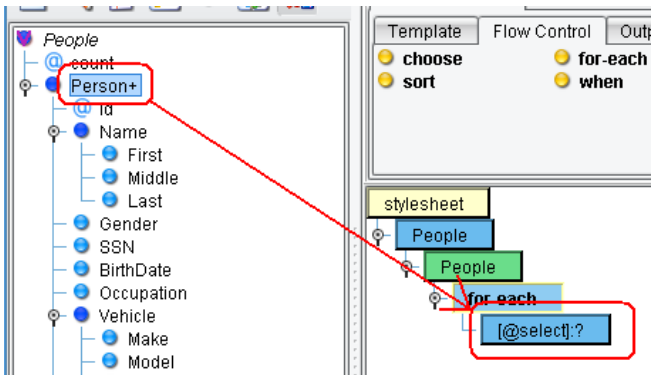
3. Drag the “for-each” instruction onto the People element from the Target in the mapping panel:



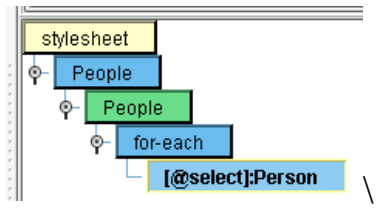
This will create a “for-each” XSLT instruction as a child of the Target People element:

The “for-each” instruction is pretty simple; it iterates over each node returned from the expression in its “select” attribute and executes each of its child instructions or elements against that node.

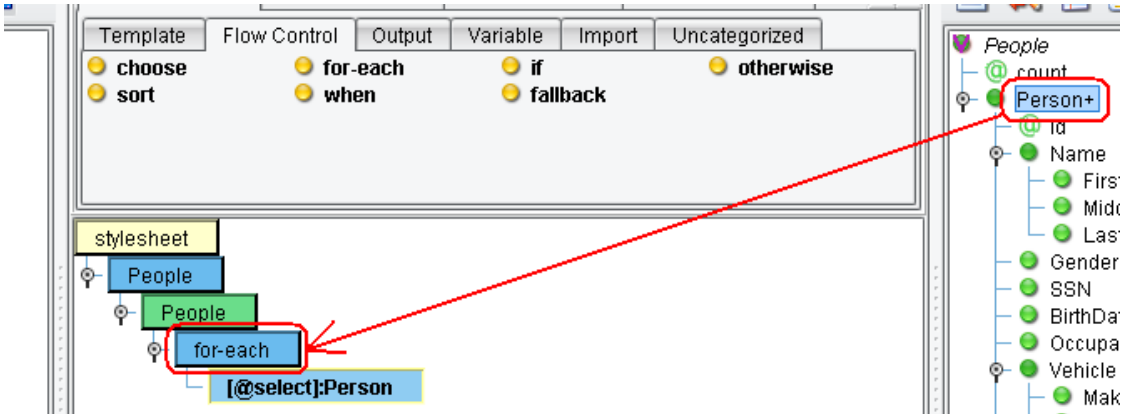
4. Iterate over each Person in the Source, so drag Person from the Source panel onto the “for-each” “@select” attribute:



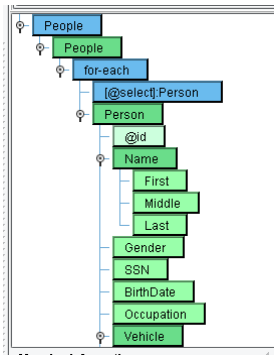
You should see the “@select” attribute change from the “?” value to a path for the “Person” element:



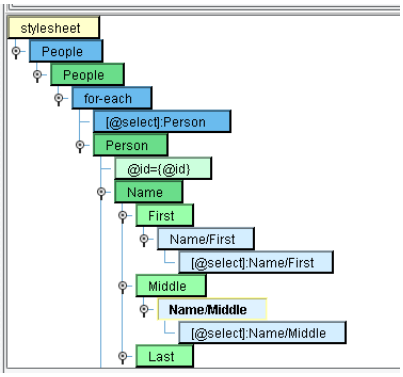
The “for-each” instruction will now execute for each Person in the Source. The action we'll want to take is to create a Person in the Target. Drag “Person” from the Target panel onto the “for-each” element:



Map the various child elements and attributes of Person from the Target onto the newly created Person in the center:



Once again, we'll map the corresponding values in the Source Person onto the Person in the center panel. There's one conceptual difference, however. This time, each expression used to populate the elements with values is relative to the for-each's current node. Because “for-each” evaluates for each Person in the Source, that means that each expression is evaluated against that Person. This means that the third iteration's expressions will be evaluated against the third Person. The Data Mapper is automatically aware of context, meaning you do not need to do anything but drag-and-drop to make use of these concepts. Map the Source Person onto the Person in the center:



Save the mapping, test it, and compare the results with the source. In this case, they should be identical.

*** Bonus ***

If you would like to take on an extra challenge try to sort the people by their State.

Hint: use the <xsl:sort> command within the <xsl:for-each>