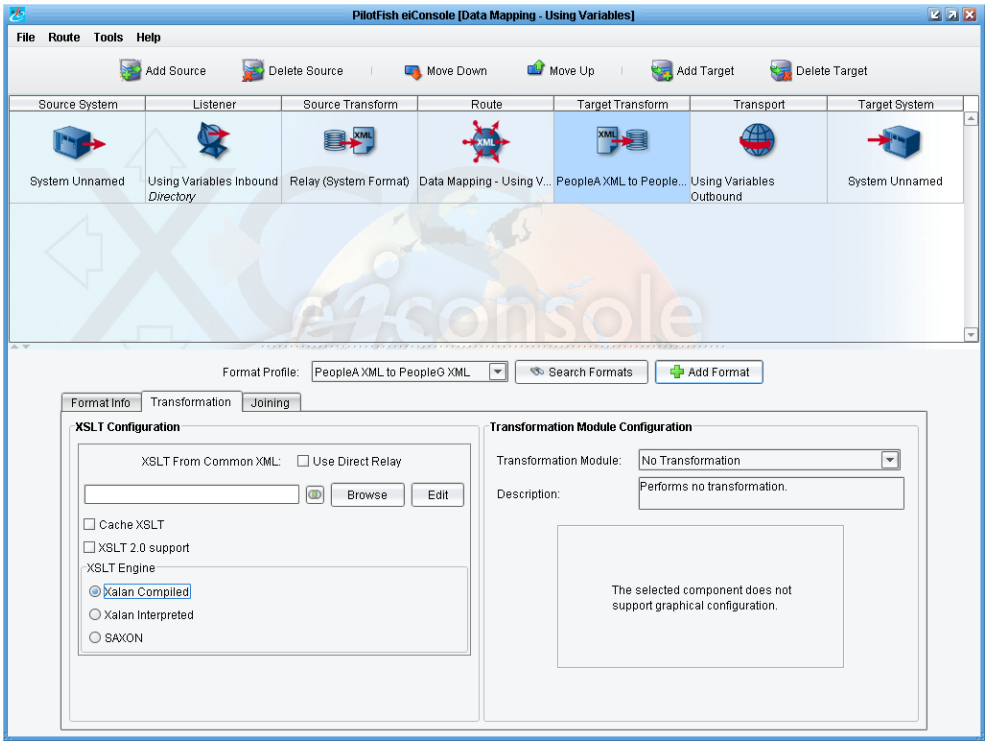
Data Mapping – Using Variables

Overview

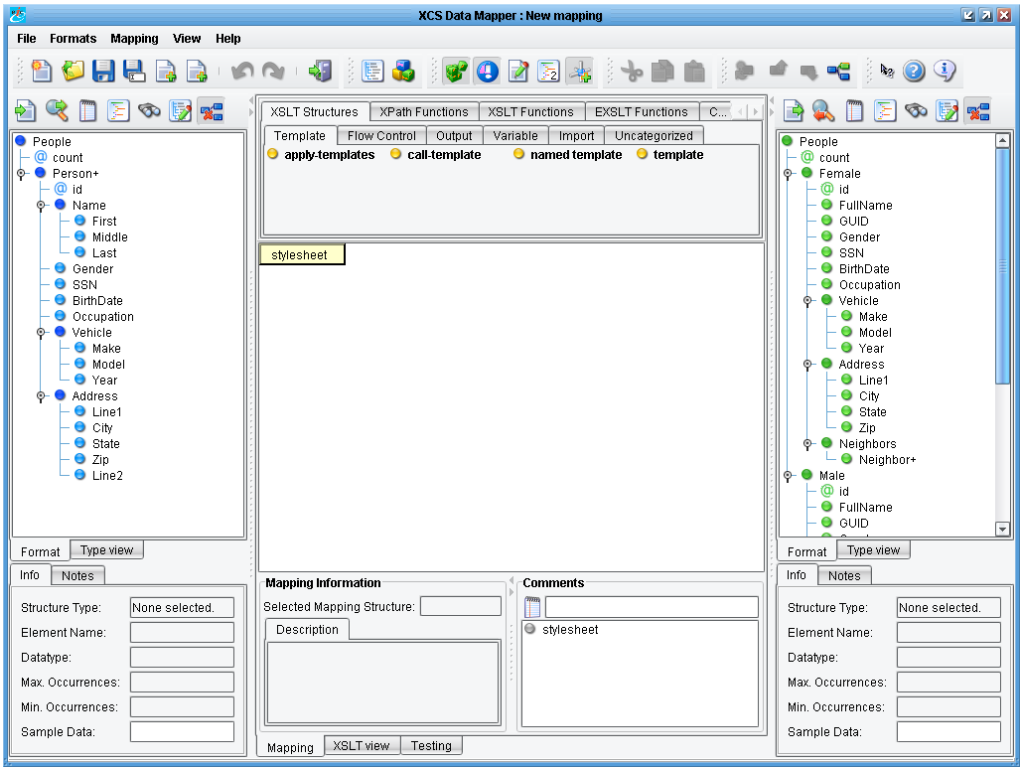
This tutorial covers the use of variables within the Data Mapper for storing and retrieving values within transformations. This tutorial expands on the concepts covered in "Data Mapping – Using Java Call-Outs," so users are expected to be familiar with that content.

Steps

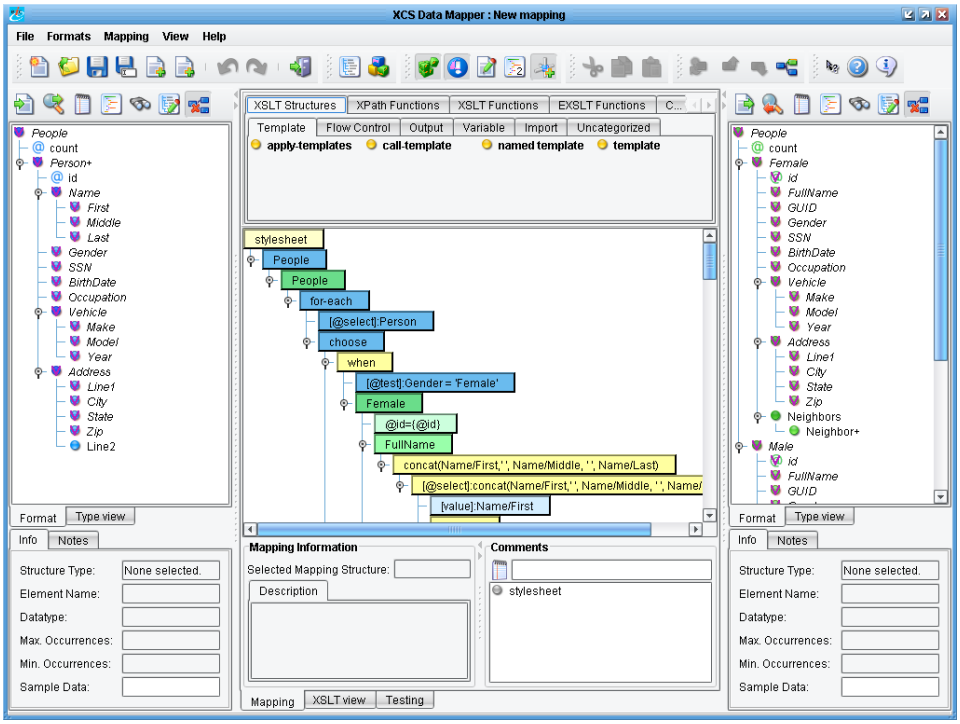
Start by creating and configuring a new Route similar to the prior tutorial's and create a new Format called "PeopleA to PeopleG" on the Target Transform stage:



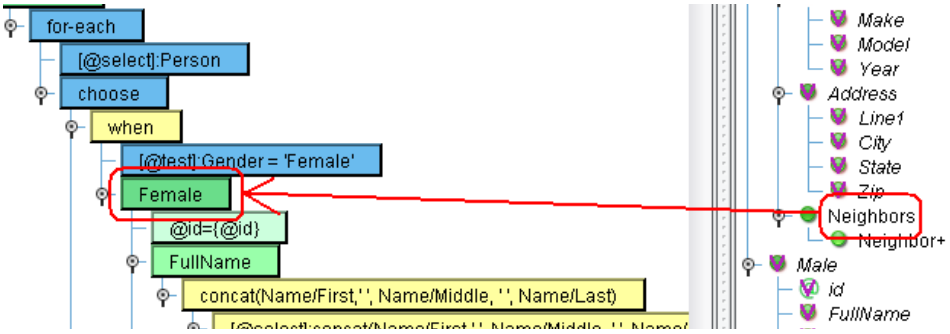
Click “Edit” to open the Data Mapper and load “PeopleA.xml” for the Source and “PeopleG.xml” for the target:



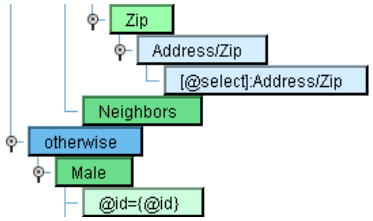
Repeat the mapping exercise from the previous tutorial or copy the resulting XSLT via the “XSLT View” tab:



Once again, a small change has been made to the Target format: the addition of the “Neighbors” element and its child “Neighbor” element. For each Person within the same State as any given Male or Female, we'll put the concatenated First, Middle, and Last Names into a Neighbor element. Start by dragging “Neighbors” onto “Female” and “Male”:

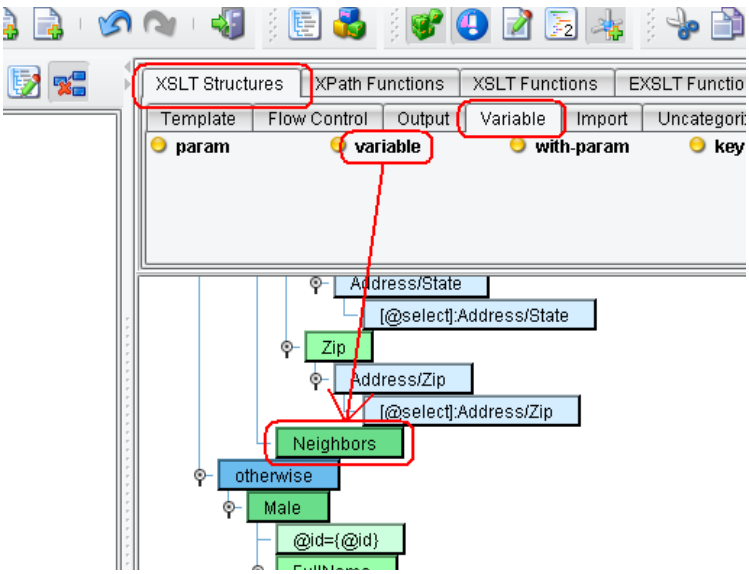


Once the Neighbor element is added, it should look like this:

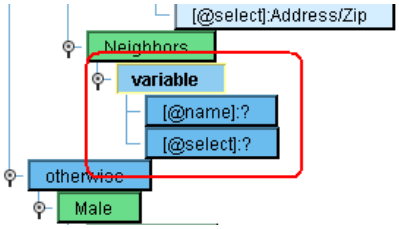


We'll be using a "for-each" and "if" instruction to iterate over each Person once again, then conditionally place a Neighbor if the State matches. However, once we place the for-each, we'll lose the context of the current Person – it will be replaced by the Person we're iterating over – as such we will need to store the state in a variable.

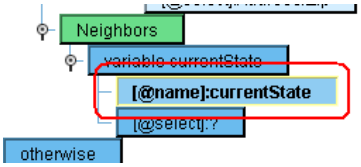
Create a variable by dragging XSLT Structures → Variable → variable onto the "Neighbors" element:



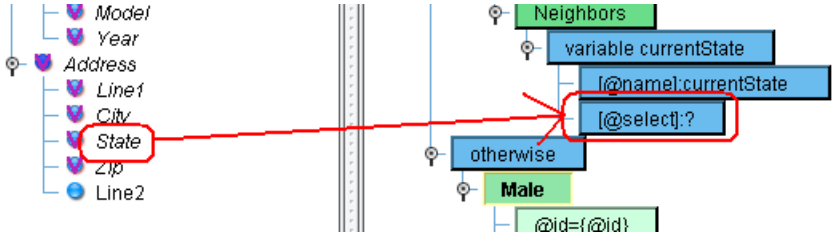
The created "variable" instruction has attributes for "name" and "select." The "name" attribute defines the name of the variable to be referenced later. The "select" attribute is an expression that will be evaluated to be placed into the variable as its value.



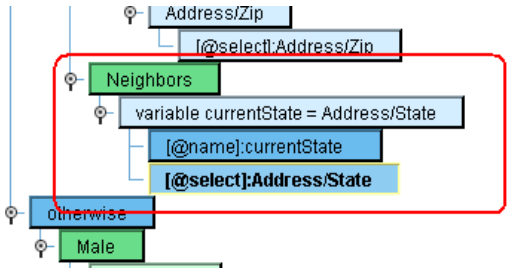
We'll set the name of our variable by double-clicking on the “name” attribute. We'll be using “currentState” for our variable name:



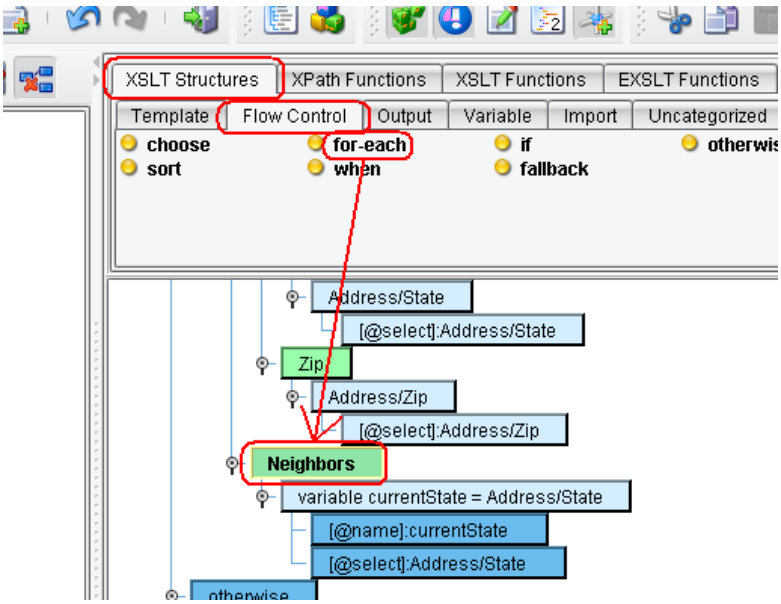
Next we need to provide a value for our variable, which in this case will be the current Person's State value. Drag-and-drop State from the Source onto the “select” attribute:



Our variable definition should now look like this:



Next we'll place our for-each onto the Neighbors element:

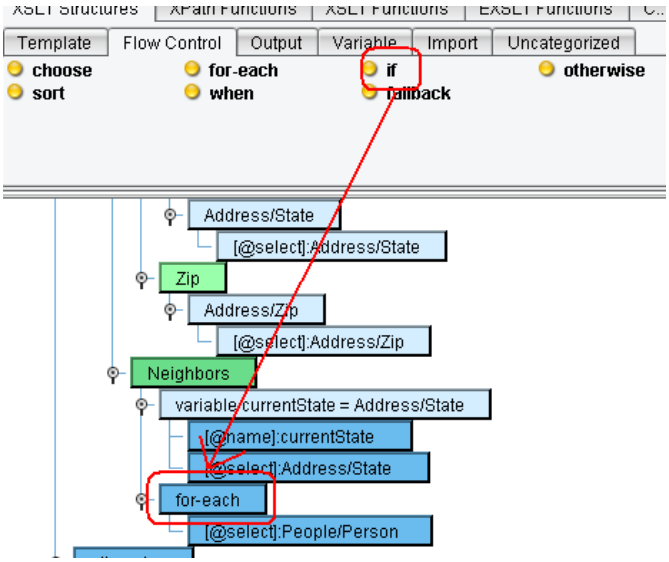


If we drag Person from the Source panel onto the for-each select, the Mapper will assume you're wanting the current Person. We'll need to manually modify the value by double-clicking it and changing it to read:

../Person

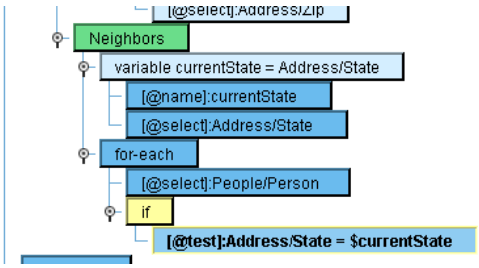
Hit enter to save the value.

Next we'll need to drag XSLT Structures → Flow Control → if onto the "for-each":

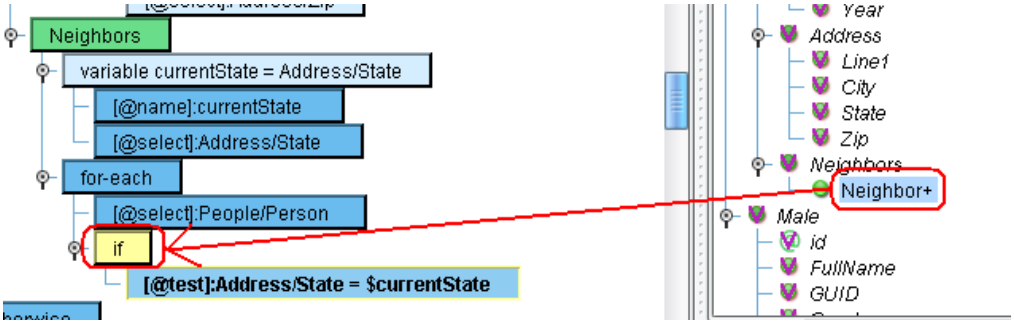


Double-click on the if "select" attribute and change it to read:

Address/State = $currentState

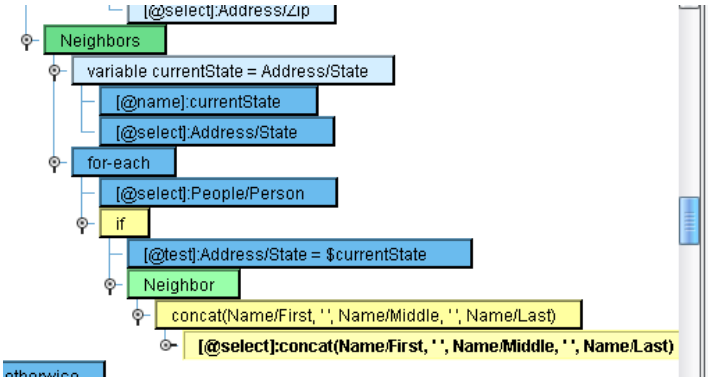


This will test to see if the current Person's State matches the previously stored State. Now we'll want to create a Neighbor element if the condition evaluates to true. Drag Neighbor from the Target panel onto the if:



Next, repeat the concatenation from the "Data Mapping – Using Functions" tutorial:

concat(Name/First,' ',Name/Last)



Finally, test the transformation:

```
<Neighbors>
    <Neighbor>Cinderella B Jacquiline</Neighbor>
    <Neighbor>Patricia U Thi</Neighbor>
    <Neighbor>Bradley E Rene</Neighbor>
</Neighbors>
```

*** Bonus ***

You may have noticed that the same Person iterating on shows in the 'neighbors' list. Try to fix this by removing them from the list. Hint: try using generate-id() function, which uniquely identifies a node within a tree.