

# Creating a Simple Mapping

## Overview

In this tutorial, we'll cover using the Data Mapper to create XSLT to map between one format of XML and another. This tutorial expands on the concepts covered in "Using Transformers," so users are expected to be familiar with that material.

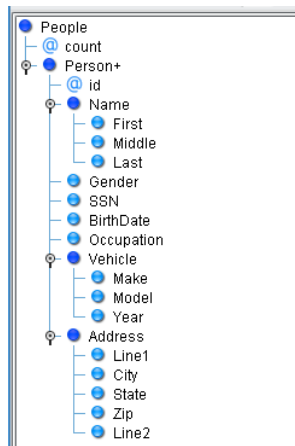
## Steps

Start by creating a new Route and configuring a Directory Listener / Transport pair (you can copy the Route created in the first lab):

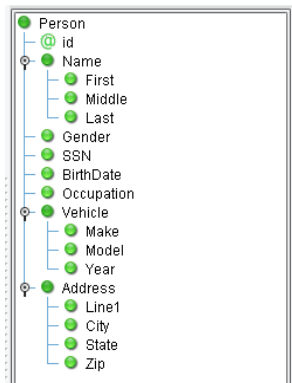
Add a new Format to the Target Transform. Call this "PeopleA to PeopleB".

1. Click "New" to open the DataMapper and create a new mapping
2. Using the "XML Format Builder", load source format "PeopleA.xml"

You should now have a Source panel that resembles this:

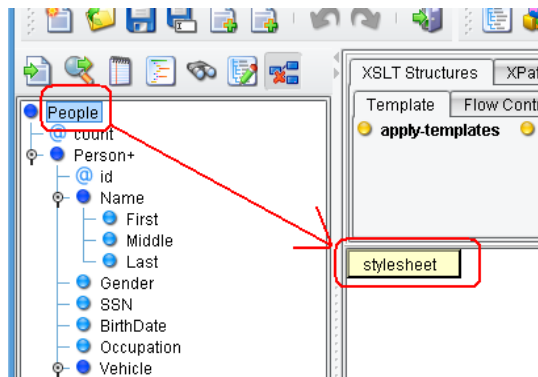


3. Load target format "PeopleB.xml"
4. Review the Target format panel:

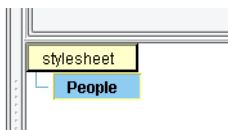


For this mapping, there's not a significant difference between the two formats except for one detail: the Source format has a “People” root element, and the “Person” elements underneath it repeat. Our Target has only the one Person. We'll simply be directly mapping the first Person in the Source to the only Person in the Target.

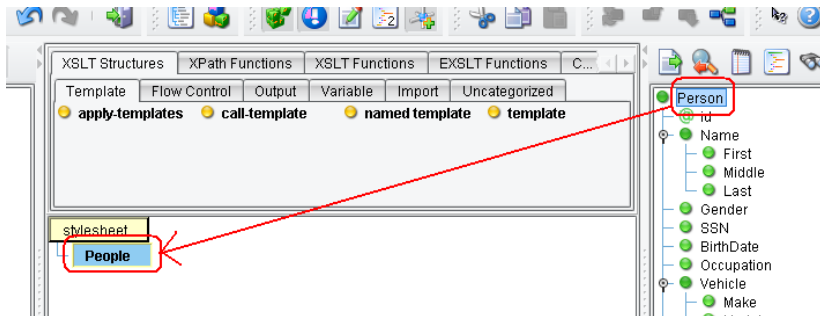
5. Select the “People” element from the Source and drag-and-drop it onto the “stylesheet” element in the center.



This will create a “People” template in the center:



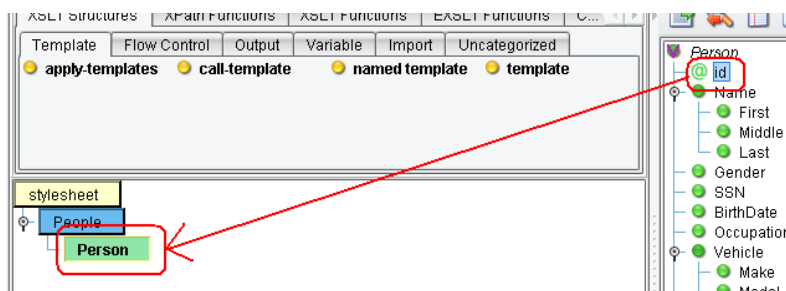
6. Drag “Person” from the Target format onto “People” in the center:

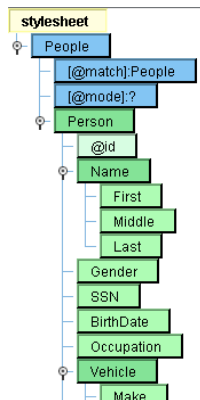


If we were to test this mapping against our Source sample, we'd get an XML output with only a “Person” element. We'll now want to create the various elements shown in the Target format.

7. Drag-and-drop each of the target elements onto the “Person” in the center. Note that child elements will need to be dragged onto the respective parents. For example, “First” should be dragged onto “Name” instead of onto “Person”:

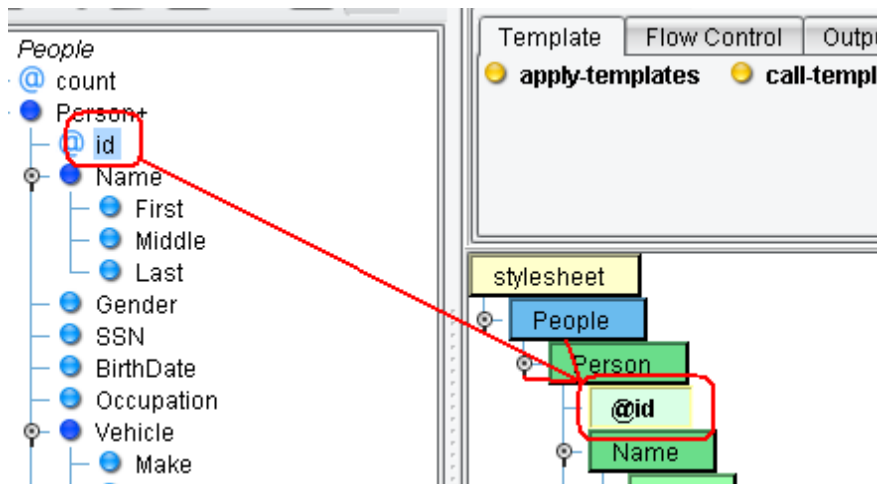
The center mapping should look exactly like this:



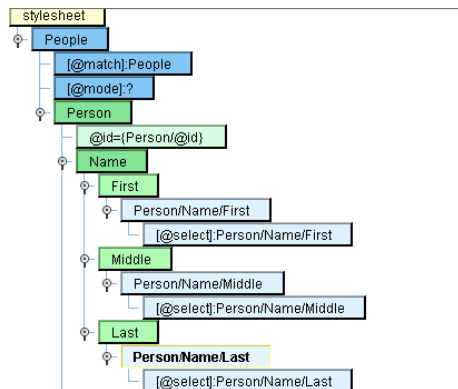


If we were to run this transformation now, we'd get a single Person element with lots of empty child elements. What we need to do now is to provide each of these with values.

8. To populate the "@id" attribute, drag the corresponding "@id" attribute from the Source format onto it:

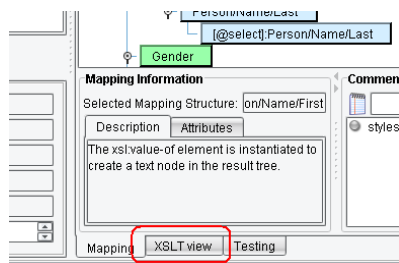


9. Continue this for each of the values. Note that you will not drag "Name" onto the center "Name" - only it's child elements (so "First" onto "First"). When you're finished, it should look something like this:



We have two other sections to investigate now.

First, click the “XSLT View” tab at the bottom of the screen:



This will change the mapping panel to show the underlying XSLT:

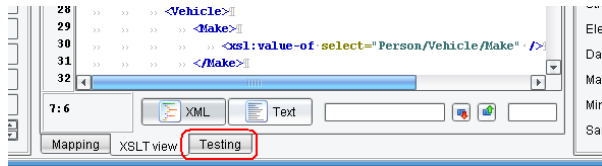
```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ?>
3   <xsl:template match="/People">
4     <Person id="(Person/@id)">
5       <Name>
6         <First>
7           <xsl:value-of select="Person/Name/First" />
8         </First>
9         <Middle>
10          <xsl:value-of select="Person/Name/Middle" />
11        </Middle>
12        <Last>
13          <xsl:value-of select="Person/Name/Last" />
14        </Last>
15      </Name>
16    </Person>
17  </template>
18 </xsl:stylesheet>

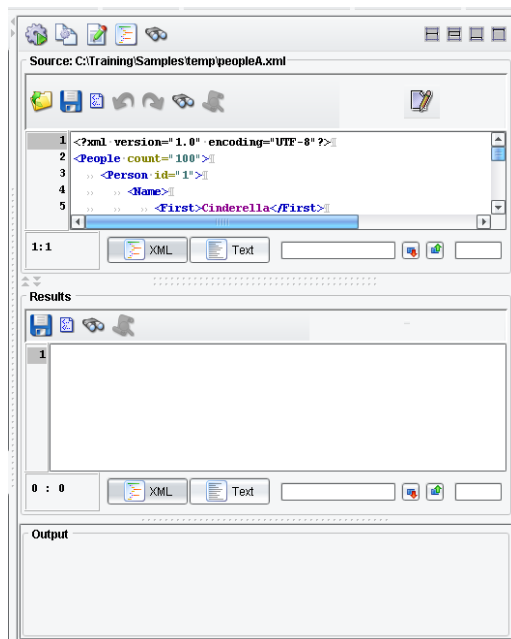
```

You can make changes directly to the XSLT view and they will show up in the Mapping (GUI) view. This particular editor features auto-completion,

auto-formatting, and a few other useful features for doing manual editing. Next, click the “Testing” tab:

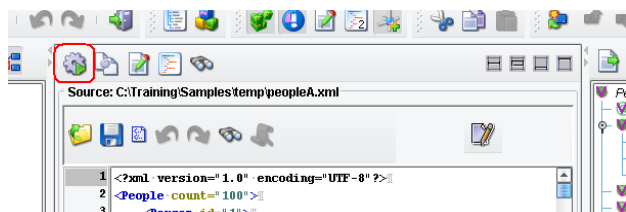


This will open the Testing mode for the Data Mapper:

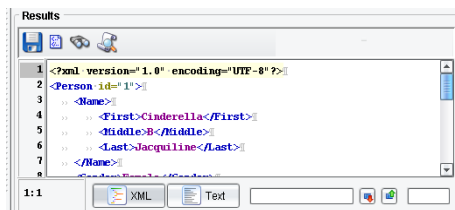


This is made up of three panels. The first is the Source Sample, which, if you used the XML Format Builder, should already have our “PeopleA.xml” file loaded for testing. The second panel is the “Results” panel, which shows the results of the transformation. Finally, the “Output” panel shows any messages from the XSLT engine, such as errors or warnings.

10. To execute the transformation, click the “Execute Transformation” button:

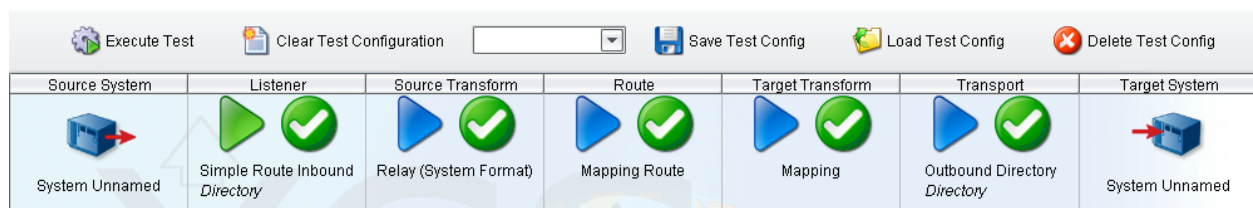


The “Results” panel will change to show the results of the transformation:

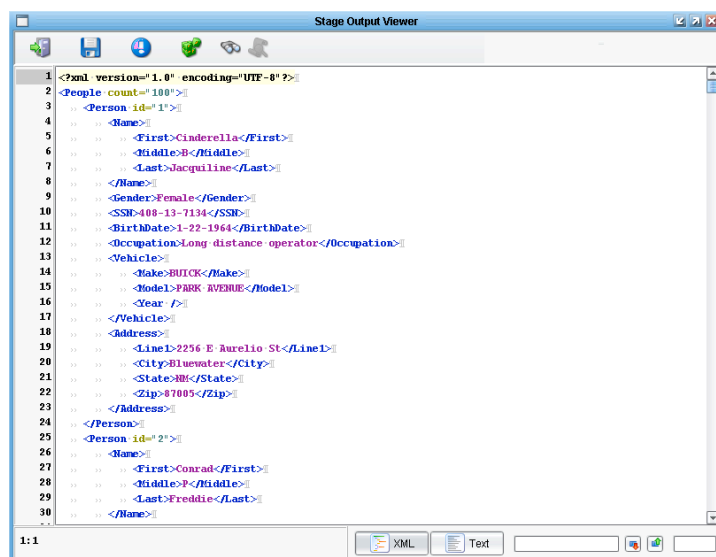


We've now completed our basic mapping.

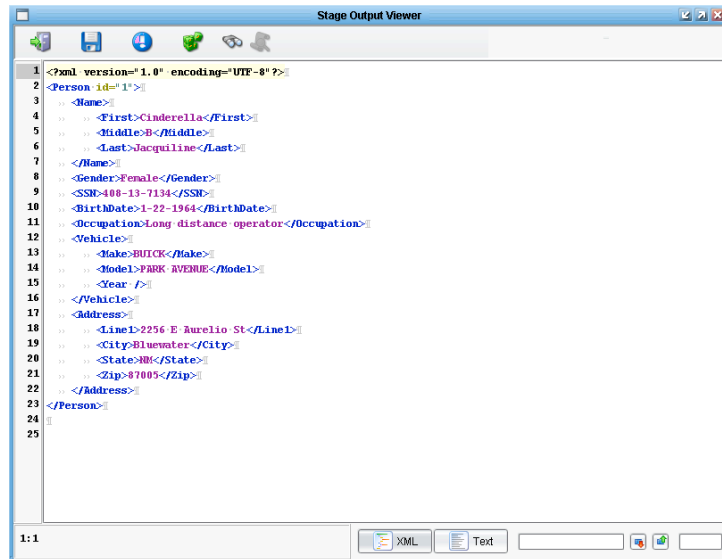
11. Save the mapping using the “Save” button or menu item (we'll name ours “Simple Mapping”) and close the Data Mapper to return to the eiConsole. From there, switch to Testing Mode, execute a test, and review the before-and-after.



Before:



And our resulting “after”:



```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <Person id="1">
3   <Name>
4     <First>Cinderella</First>
5     <Middle>B</Middle>
6     <Last>Jacqueline</Last>
7   </Name>
8   <Gender>Female</Gender>
9   <SSN>444-13-7134</SSN>
10  <BirthDate>1-22-1964</BirthDate>
11  <Occupation>Long distance operator</Occupation>
12  <Vehicle>
13    <Make>BUICK</Make>
14    <Model>PARK AVENUE</Model>
15    <Year />
16  </Vehicle>
17  <Address>
18    <Line1>2256 E Aurelio St</Line1>
19    <City>Bluewater</City>
20    <State>MI</State>
21    <Zip>48005</Zip>
22  </Address>
23 </Person>
24
25

```

As you can see, performing mappings is simply a matter of dragging and dropping between the Source and Target formats. In our next tutorial, we'll cover increasing the complexity and functionality a bit through iteration.