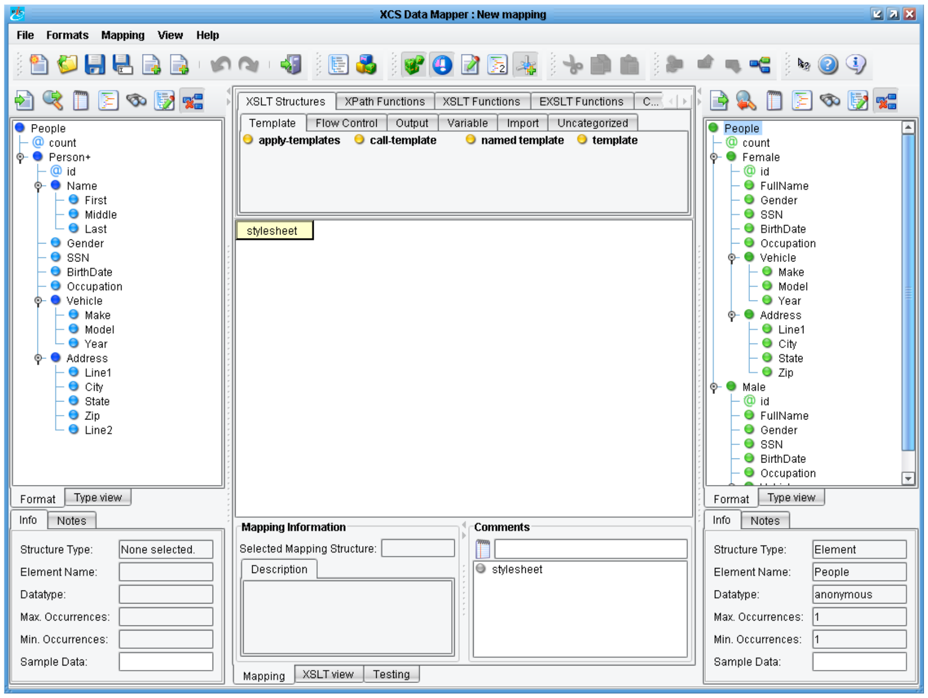
Advanced Mapping – Using Tabular Mappings

Overview

In this tutorial we'll cover using the “Tabular Mappings” feature of the Data Mapper to convert between Source and Target code sets. This tutorial expands on concepts covered in “Data Mapping – Using Functions,” so users are expected to be familiar with that content.

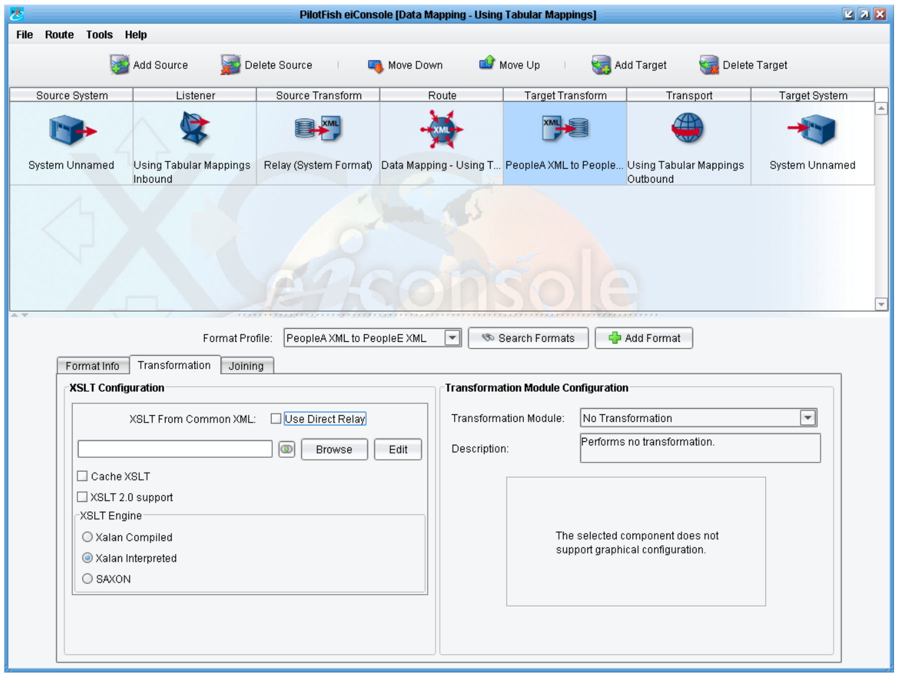
Steps

Start by creating and configuring a new Route similar to the prior tutorials. Add a new Format named “PeopleA to PeopleE” on the Target Transform, and uncheck “Use Direct Relay”:



Structure Type:	None selected.
Element Name:	
Datatype:	
Max. Occurrences:	
Min. Occurrences:	
Sample Data:	

Structure Type:	Element
Element Name:	People
Datatype:	anonymous
Max. Occurrences:	1
Min. Occurrences:	1
Sample Data:	



Format Profile: PeopleA XML to PeopleE XML

XSLT Configuration

XSLT From Common XML: ☐ Use Direct Relay

☐ Cache XSLT
☐ XSLT 2.0 support

XSLT Engine
○ Xalan Compiled
● Xalan Interpreted
○ SAXON

Transformation Module Configuration

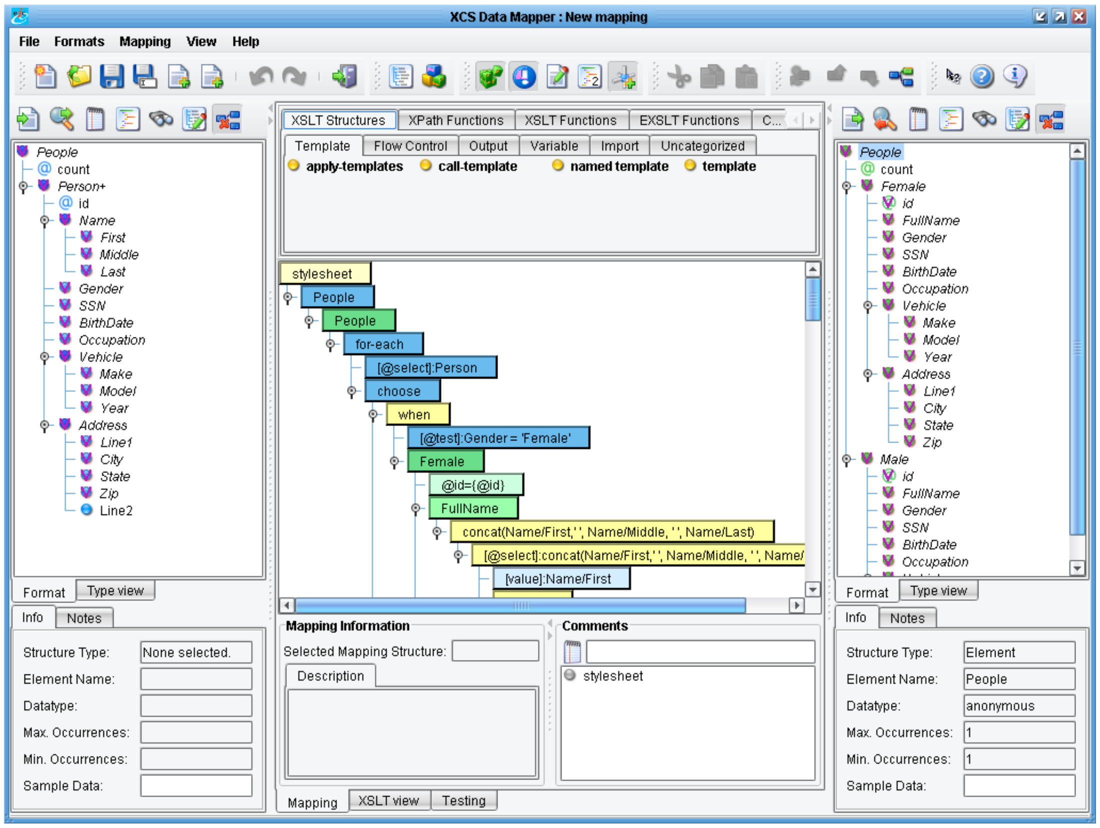
Transformation Module: No Transformation

Description: Performs no transformation.

The selected component does not support graphical configuration.

Click "Edit" to open the Data Mapper. Load "PeopleA.xml" for the Source and "PeopleE.xml" for the target:

Repeat the mapping exercise from the previous tutorial, or copy the XSLT using the "XSLT View" tab:



```
People
  count
  Person+
    id
    Name
      First
      Middle
      Last
    Gender
    SSN
    BirthDate
    Occupation
    Vehicle
      Make
      Model
      Year
    Address
      Line1
      City
      State
      Zip
      Line2
```

```
stylesheet
  People
    People
      for-each
        [@select]:Person
        choose
          when
            [@test]:Gender = 'Female'
            Female
              @id={@id}
              FullName
                concat(Name/First,' ', Name/Middle, ' ', Name/Last)
                  [@select]:concat(Name/First,' ', Name/Middle, ' ', Name/
                    [value]:Name/First
```

```
People
  count
  Female
    id
    FullName
    Gender
    SSN
    BirthDate
    Occupation
    Vehicle
      Make
      Model
      Year
    Address
      Line1
      City
      State
      Zip
  Male
    id
    FullName
    Gender
    SSN
    BirthDate
    Occupation
```

Structure Type:	None selected.
Element Name:	
Datatype:	
Max. Occurrences:	
Min. Occurrences:	
Sample Data:	

Mapping Information

Selected Mapping Structure:

Description

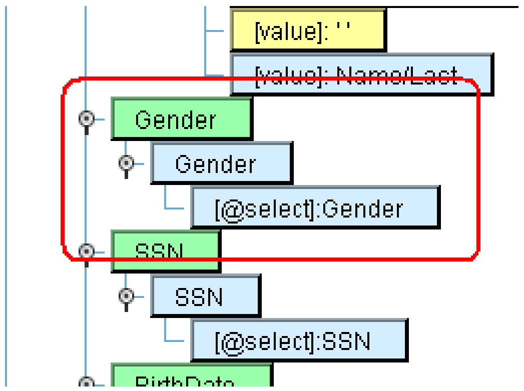
Comments

stylesheet

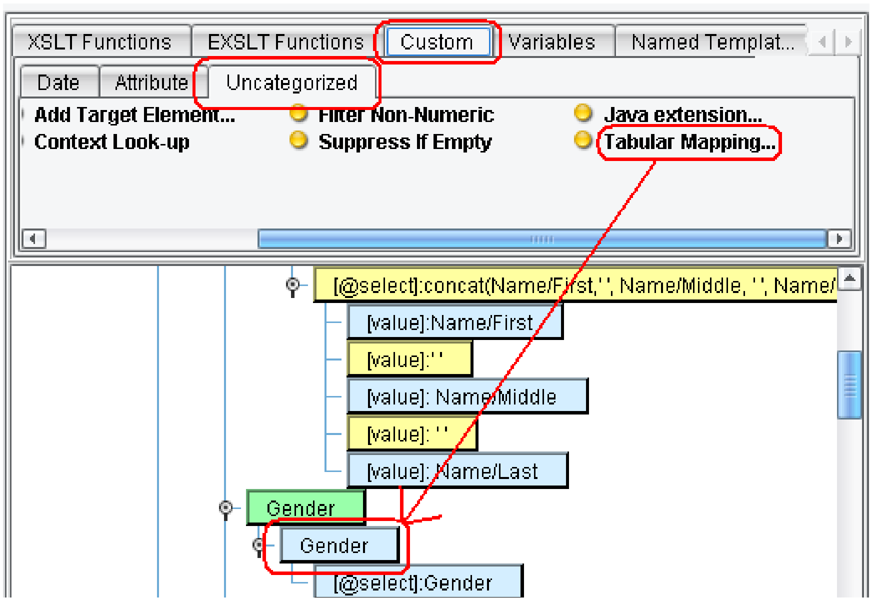
Structure Type:	Element
Element Name:	People
Datatype:	anonymous
Max. Occurrences:	1
Min. Occurrences:	1
Sample Data:	

This mapping is pretty much identical to the last except for one detail: in the Source format, Gender is represented by “Male” or “Female,” while in the Target, Gender is represented by “M” or “F.” Most formats that make use of similar, coded values do not share entry values. For example, one format might represent marital status with “Single” or “Divorced,” while in another those respective values could be “1” and “6.” While you can build out these conversions using the “choose / when / otherwise” instructions, there's a much simpler way: Tabular Mappings.

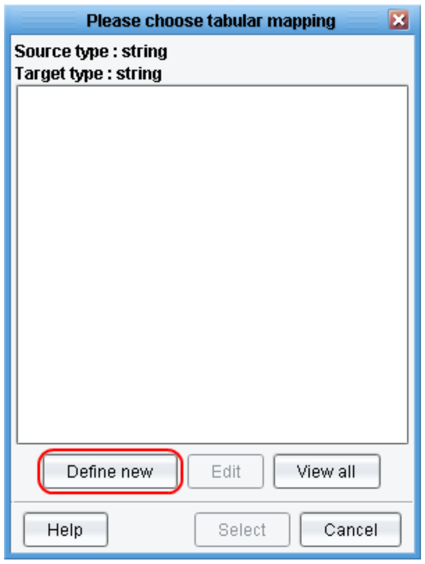
Scroll down to the Gender mapping in the panel:



You can see that Gender gets its value from Gender in the Source. To apply a Tabular Mapping to this mapping, drag “Tabular Mapping” from Custom → Uncategorized onto it:

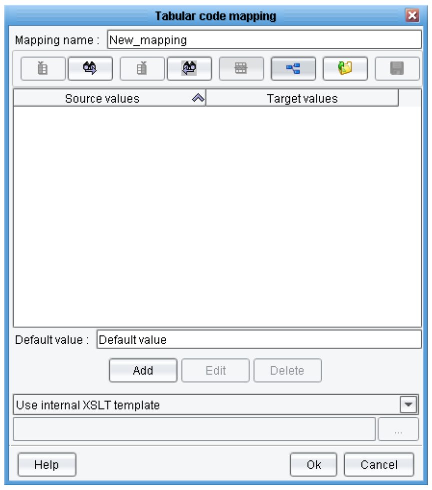


This will open the Tabular Mapping dialog, where you can select from a list of existing mappings:



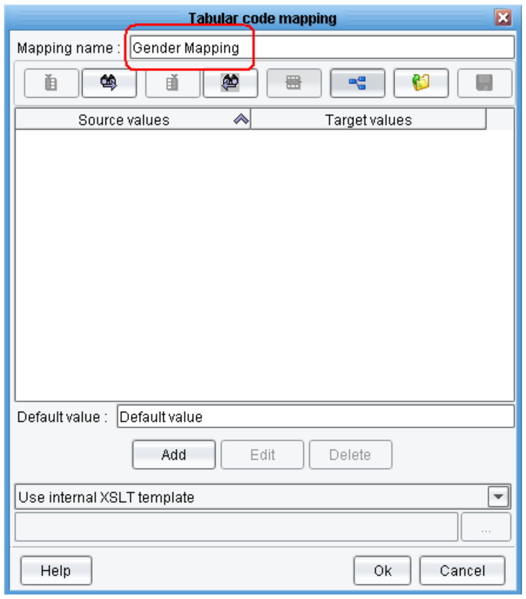
Source type : string
Target type : string

Of course, we have no such mappings defined yet, so click "Define New" to open a dialog to create a new Tabular Mapping:

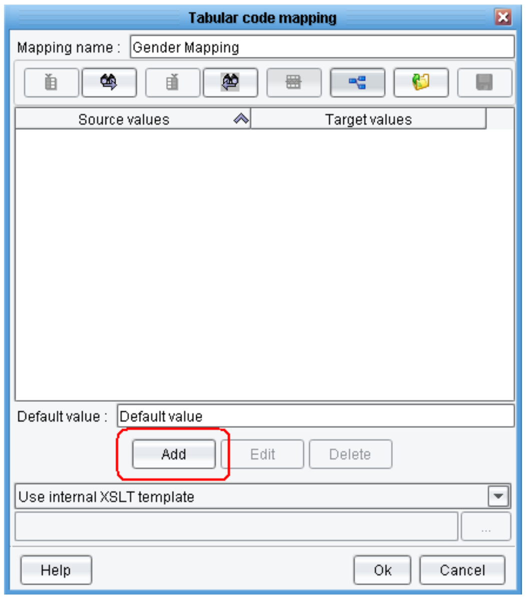


Mapping name : New_mapping
Default value : Default value
Use internal XSLT template

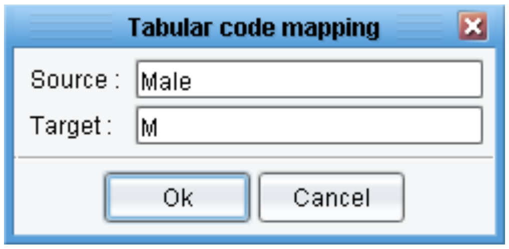
This dialog provides a field to name the mapping as well as a table of currently mapped values. If the Source or Target formats loaded in provide enumerations (such as from an XSD or certain industry-specific standards) then the Source and / or Target value columns will have rows pre-populated. As this is not the case for our sample, we'll have to manually add rows. We'll start by naming our mapping:


Tabular code mapping
Mapping name : Gender Mapping
Source values
Target values
Default value : Default value
Add
Edit
Delete
Use internal XSLT template
Help
Ok
Cancel

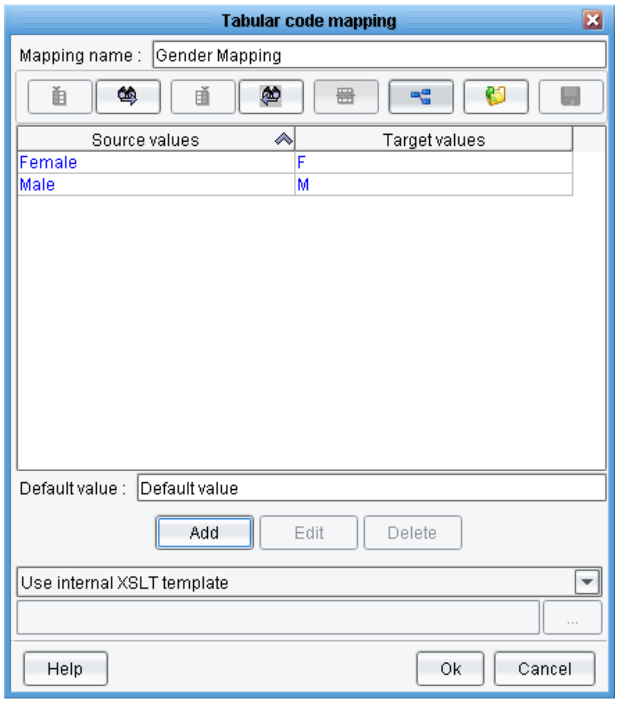
Next, click the "Add" button to define a new row:


Tabular code mapping
Mapping name : Gender Mapping
Source values
Target values
Default value : Default value
Add
Edit
Delete
Use internal XSLT template
Help
Ok
Cancel

The raised dialog will provide fields to enter the Source and Target values. Since we're converting between “Male” / “Female” and “M” / “F” codes, provide “Male” and M” for the first row:



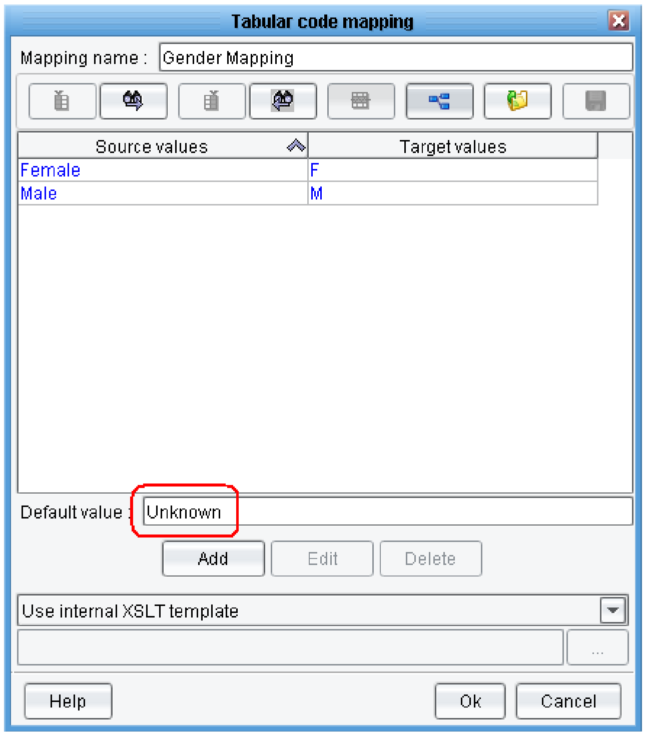
Click “OK,” then click “Add” again and do the same for “Female” and “F.” You should now have a table with two rows:


Tabular code mapping
Mapping name : Gender Mapping

Source values	Target values
Female	F
Male	M

Default value : Default value
Add
Edit
Delete
Use internal XSLT template
...
Help
Ok
Cancel

Next we'll need to define a “Default Value” - the value that is chosen if the Source “Gender” is neither “Female” nor “Male.” We'll default to “Unknown”:

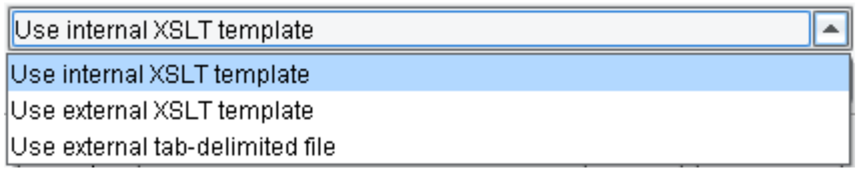

Tabular code mapping

Mapping name : Gender Mapping

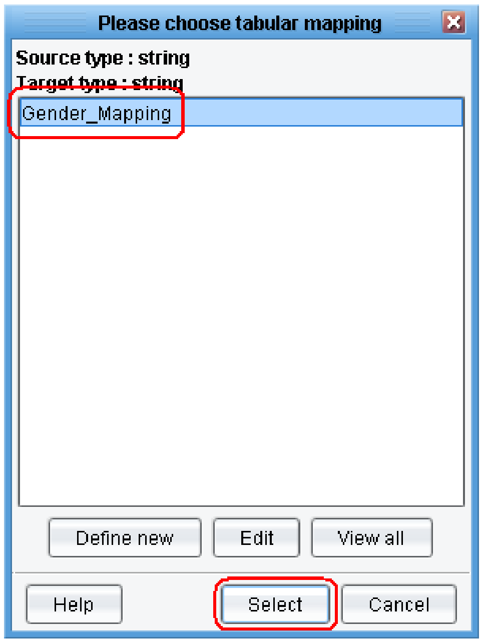
Source values	Target values
Female	F
Male	M

Default value : Unknown

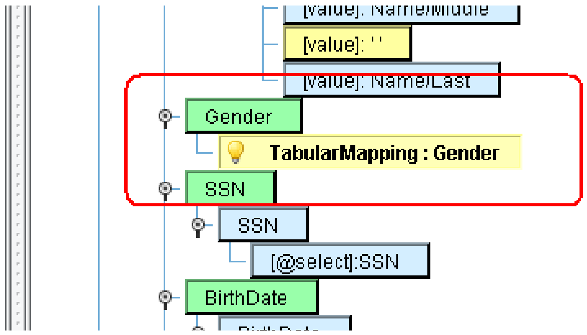
Notice that we can save the “Tabular Mapping” internally (default), externally or as a tab delimited flat file:


Use internal XSLT template
Use internal XSLT template
Use external XSLT template
Use external tab-delimited file

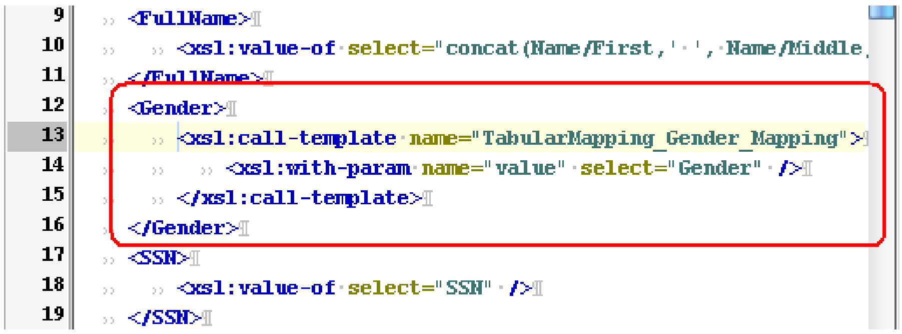
Finally, click “OK,” select the new mapping in the previous dialog, and click “Select”:



The resulting mapping element should now look like this:

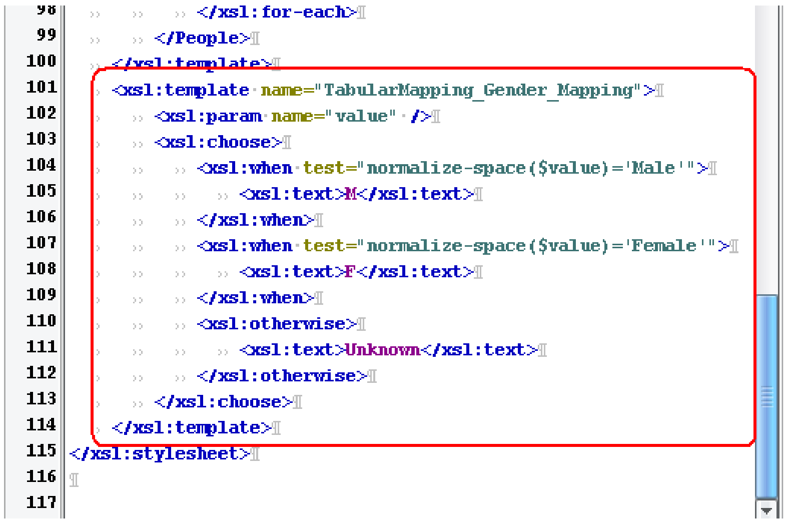


Finally, drop into the XSLT view to see how the Tabular Mapping is handled:



```
<FullName>
    <xsl:value-of select="concat(Name/First,' ',Name/Middle,
</FullName>
<Gender>
    <xsl:call-template name="TabularMapping_Gender_Mapping">
        <xsl:with-param name="value" select="Gender" />
    </xsl:call-template>
</Gender>
<SSN>
    <xsl:value-of select="SSN" />
</SSN>
```

If you scroll further down, you can see the implementation of the “template” which is called here to see the choose / when / otherwise logic that the Tabular Mapping tool generated automatically:

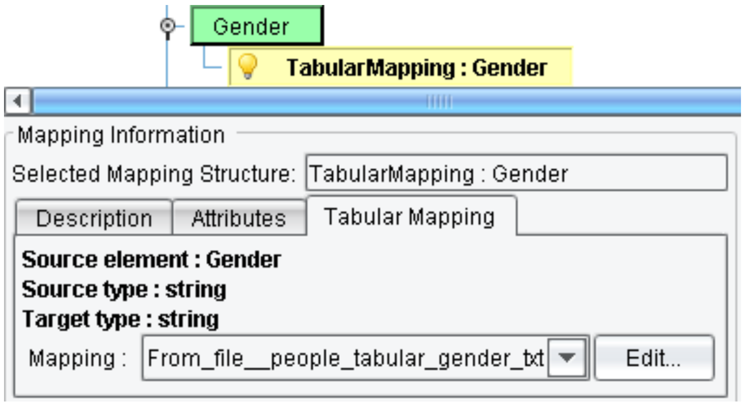


```
        </xsl:for-each>
    </People>
</xsl:template>
<xsl:template name="TabularMapping_Gender_Mapping">
    <xsl:param name="value" />
    <xsl:choose>
        <xsl:when test="normalize-space($value)='Male'">
            <xsl:text>M</xsl:text>
        </xsl:when>
        <xsl:when test="normalize-space($value)='Female'">
            <xsl:text>F</xsl:text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>Unknown</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

As you can see, Tabular Mappings are a powerful tool for handling conversion between code lists and values, and can save you considerable amounts of time as you tackle larger mappings.

*** Bonus ***

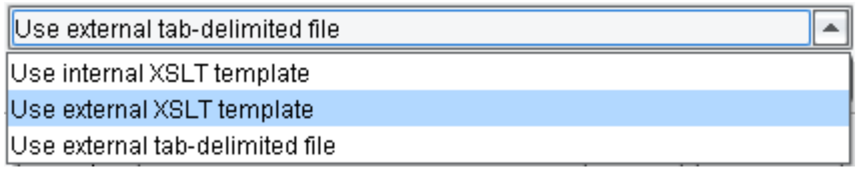
Select the ‘TabularMapping’ node in the tree and open the ‘Tabular Mapping’ tab in the ‘Mapping Information’ section at the bottom:



Selected Mapping Structure: TabularMapping : Gender

Source element : Gender
Source type : string
Target type : string
Mapping : From_file__people_tabular_gender_txt

Edit the mapping to use an 'External XSLT template' and review the results. Then edit it to 'External tab-delimited file' and note the changes. For the time being enter a fully qualified path, which can be changed to a relative path later.


Use external tab-delimited file
Use internal XSLT template
Use external XSLT template
Use external tab-delimited file

A java extension is used to load the external tab-delimited file (more to come on this in the next lab). The advantage of using this type of tabular mapping is that the code translations can be edited by anyone.