In this course you will be using <oXygen/> XML Editor version 12.3 (oXygen for short from now on) for XML related work. The work includes writing XML Schema files with corresponding XML files, writing XQuery code and running it, and performing XSLT transformations.
We will now go through the above listed tasks and see how they're done in oXygen. This is not a tutorial about XML and XML related technologies in themselves, but rather how you work with the things just mentioned in oXygen.

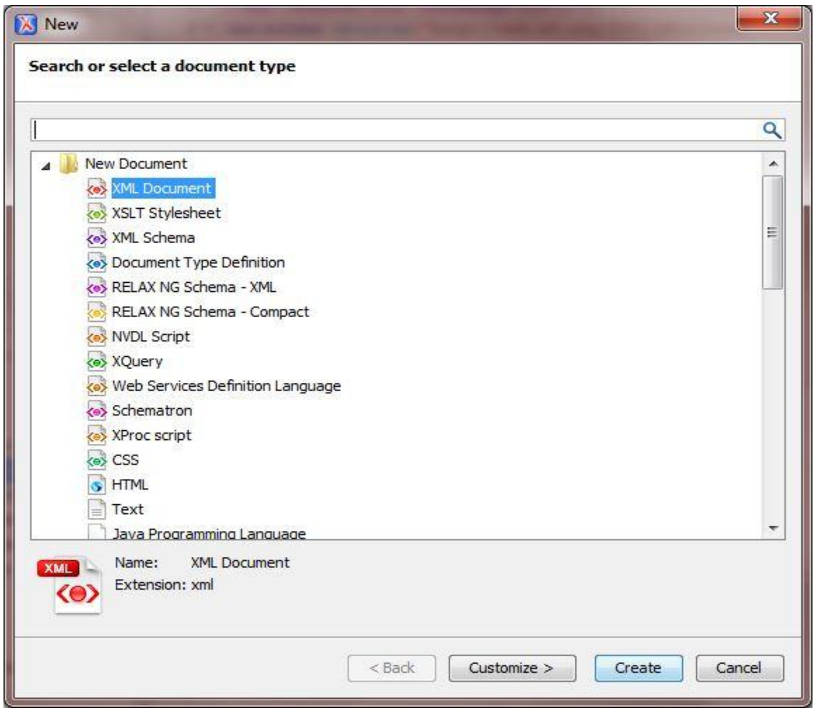
If this is your first time working with XML it might be a good idea to try and follow the tutorial actively and not just simply reading or skimming through it.

1. XML Schemas and XML data files

Writing XML Schemas and XML data files is a pleasant task in oXygen compared to a dumb text editor like Notepad for instance. Not only does oXygen assist with automatic syntax coloring and indentation, it also provides useful auto-completions, suggestions etcetera.

1.1 Create XML Schema

To create a new XML Schema or XML file you simply select the menu item ”File->New...” which brings up a dialog (picture below) where you can choose the appropriate type. From this dialog you can also create files for XQuery and XSL (these are all raw text files so, basically, you could do everything manually if you wanted, but oXygen can assist with skeleton content if you use the dialog).



Name: XML Document
Extension: xml

When you chose XML schema and create a schema you should end up with a skeleton (see below) which enables you to start defining the specific structure of the schema right away.

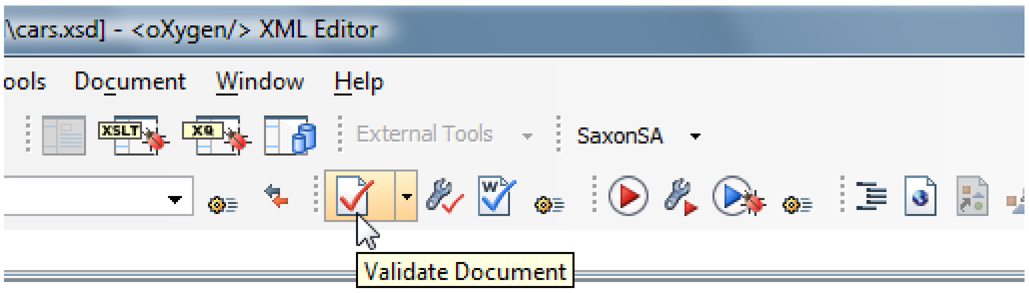
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

</xs:schema>
```

Now you can write your code or schema. In the following we will use the simple bellow saved as cars.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="cars" type="rootType"/>

  <xs:complexType name="rootType">
    <xs:sequence>
      <xs:element name="car" type="carType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="carType">
    <xs:sequence>
      <xs:element name="maker" type="xs:string"/>
      <xs:element name="model" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

1.2 Validate Schema

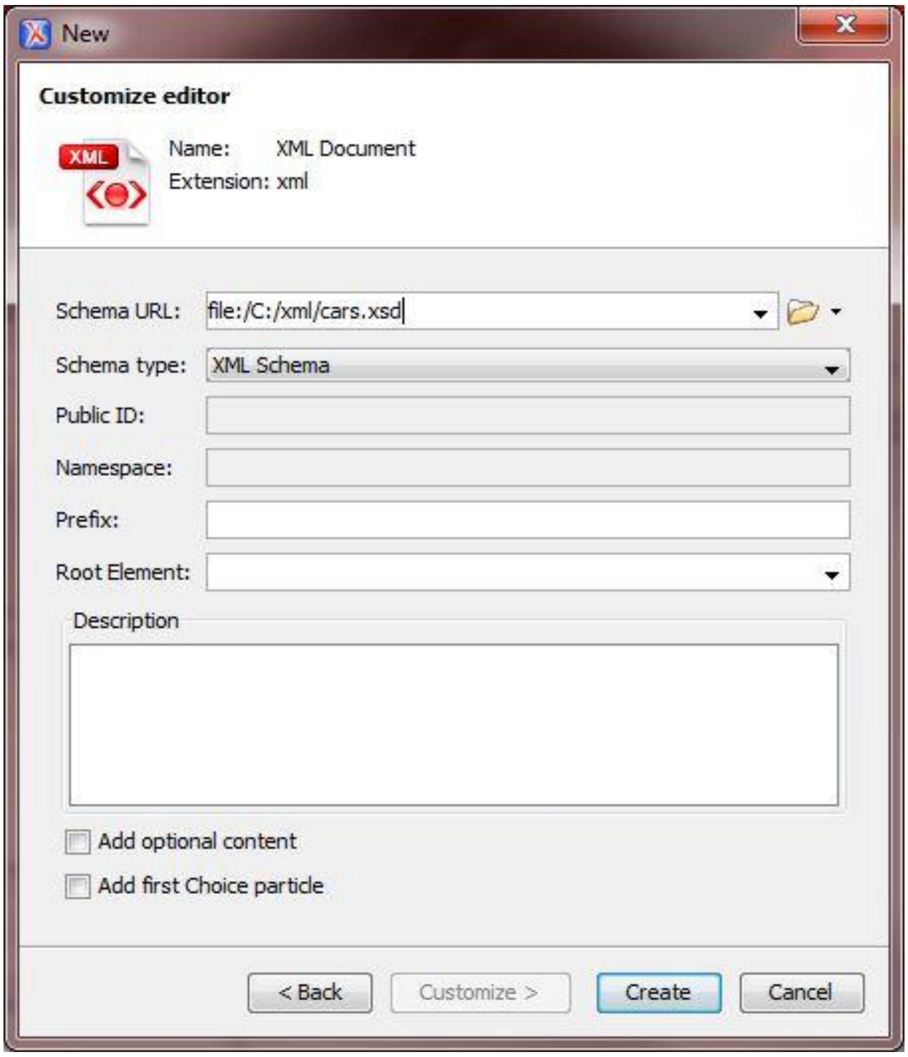
If you want to validate this schema you can press the toolbar button ”Validate Document” (displayed below) or access the menu item ”Document->Validate->Validate Document” or press Ctrl+Shift+V. However, oXygen performs on-the-fly validation so you won't need to do it manually very often. If you have several files open (oXygen can have many files open in tabs), the correct one must have focus.



Any validation errors will be underlined in the document itself and when you press the Validate Document button a window will appear in the bottom area with a summary of all errors (if there were any).

1.3 Create XML File

Now when we have a simple but complete schema and can create a corresponding XML file. Using the dialog ”New” described earlier we create an XML file. By clicking customize in the window above we can specify a schema for the document, which is convenient, because our skeleton will be more complete. The dialog below illustrates how to perform the schema association.



Name: XML Document
Extension: xml

Schema URL: file:/C:/xml/cars.xsd
Schema type: XML Schema
Public ID:
Namespace:
Prefix:
Root Element:
Description

☐ Add optional content
☐ Add first Choice particle

Below is what the skeleton will look like if we create a new XML file with the cars.xsd-schema associated.

```
<?xml version="1.0" encoding="UTF-8"?>
<cars xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="file:/C:/oxygen-tutorial/cars.xsd">
   <car>
      <maker></maker>
      <model></model>
   </car>
</cars>
```

This skeleton is now ready to be filled with data and you will notice that oXygen knows the order of the elements which enables it to perform a lot of auto-completions.

However, as you can see the file URL is specified using an absolute path which is not very convenient if you move the files between different computers and/or users. If we save this file as cars.xml and keep it together with the schema we can change the URL to:

```
xsi:noNamespaceSchemaLocation="file:cars.xsd"
```

If you were using namespaces you would be setting xsi:schemaLocation instead and the syntax would be slightly different.

2. XQuery

We continue on our cars example, say the file cars.xml now contains:

```
<?xml version="1.0" encoding="UTF-8"?>
<cars xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="file:cars.xsd">
   <car>
      <maker>BMW</maker>
      <model>750Li</model>
   </car>

   <car>
      <maker>BMW</maker>
      <model>M5</model>
   </car>
   <car>
      <maker>Volvo</maker>
      <model>V70</model>
   </car>
   <car>
      <maker>Ford</maker>
      <model>F150</model>
   </car>
</cars>
```

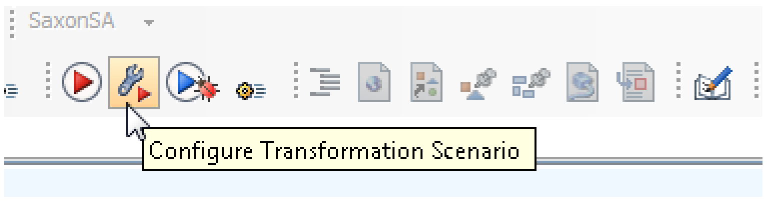
2.1 Create XQuery

We will write an Xquery that returns how many cars there are where the maker is BMW. To create an xquery-file you use the now familiar dialog ”New”. This will give you an Untitled file with the extension xquery that is empty. Save it as num_bmw.xquery in the same directory as cars.xsd and cars.xml. Write the code:

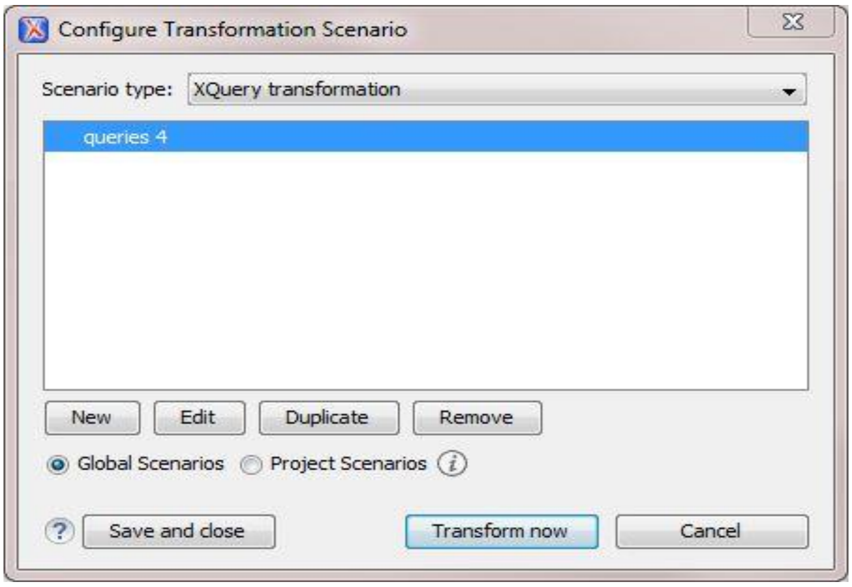
```
<res>Num cars by BMW: {count(doc('cars.xml')//car[maker='BMW'])}</res>
```

2.2 Configure oXygen for XQuery Execution

Now that we have the code for the query, it's time to run it. Make sure the XQuery file has focus and press the button ”Configure Transformation Scenario”.



A dialog appears where you don't have to configure much except select ”Execute XQuery” and press OK.

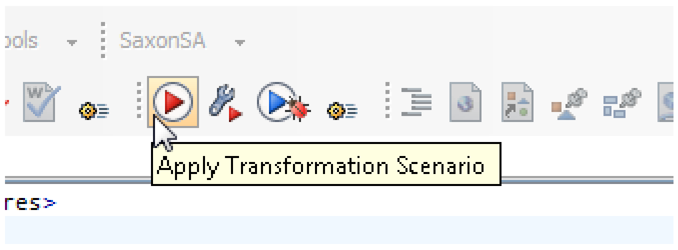


Scenario type: XQuery transformation

queries 4

2.3 Run XQuery

Now that the transformation scenario has been configured, it's time to run it. Press the ”Apply Transformation Scenario” button:

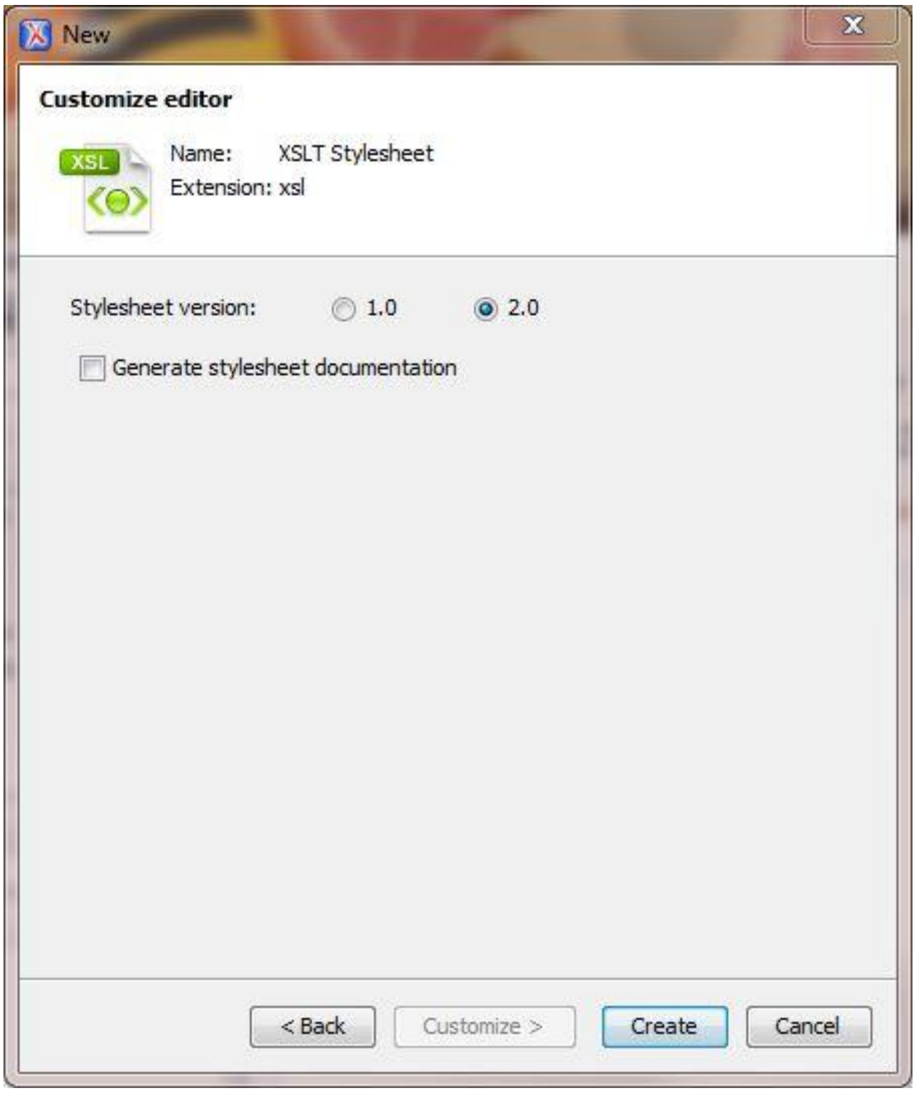


When you press it the XQuery should be executed and the result displayed at the bottom:

```
<?xml version="1.0" encoding="UTF-8"?>
<res>Num cars by BMW: 2</res>
```

3. Create XSL Transformation

Now we want to write an XSL transformation that displays maker and model in a nice HTML table. Open the dialog ”New”, select ”XSL Stylesheet” and press ”customize”. A dialog appears asking you which version you want. You should pick version 2.0 and you can also tell oXygen not to ask you again.



Name: XSLT Stylesheet
Extension: xsl

Stylesheet version: 1.0 [✓ 2.0]

☐ Generate stylesheet documentation

The following skeleton should be created in a file named Untitled.xsl.

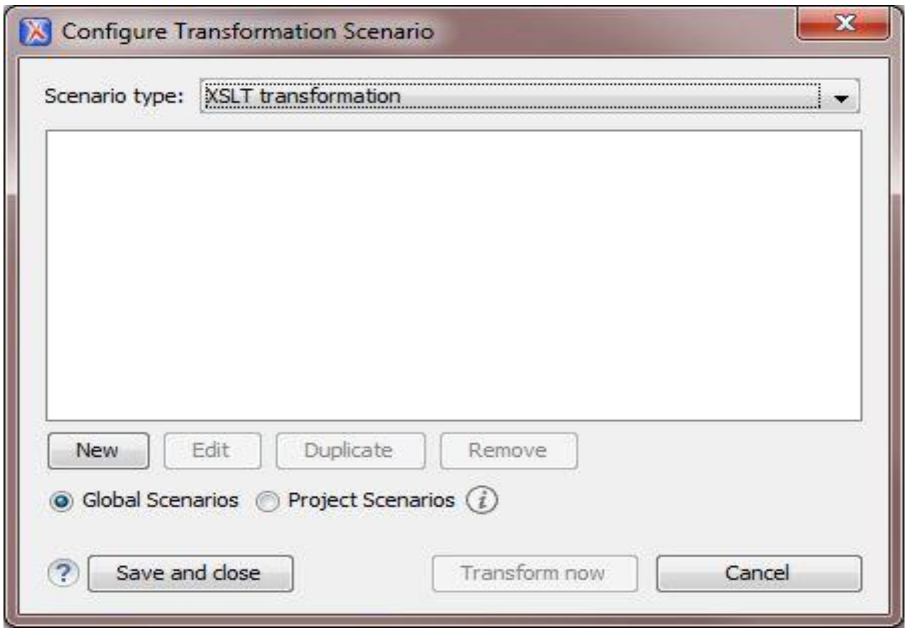
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

</xsl:stylesheet>
```

With the skeleton as a starting point, we code the following XSL transformation which should create a sorted table of makers and model:

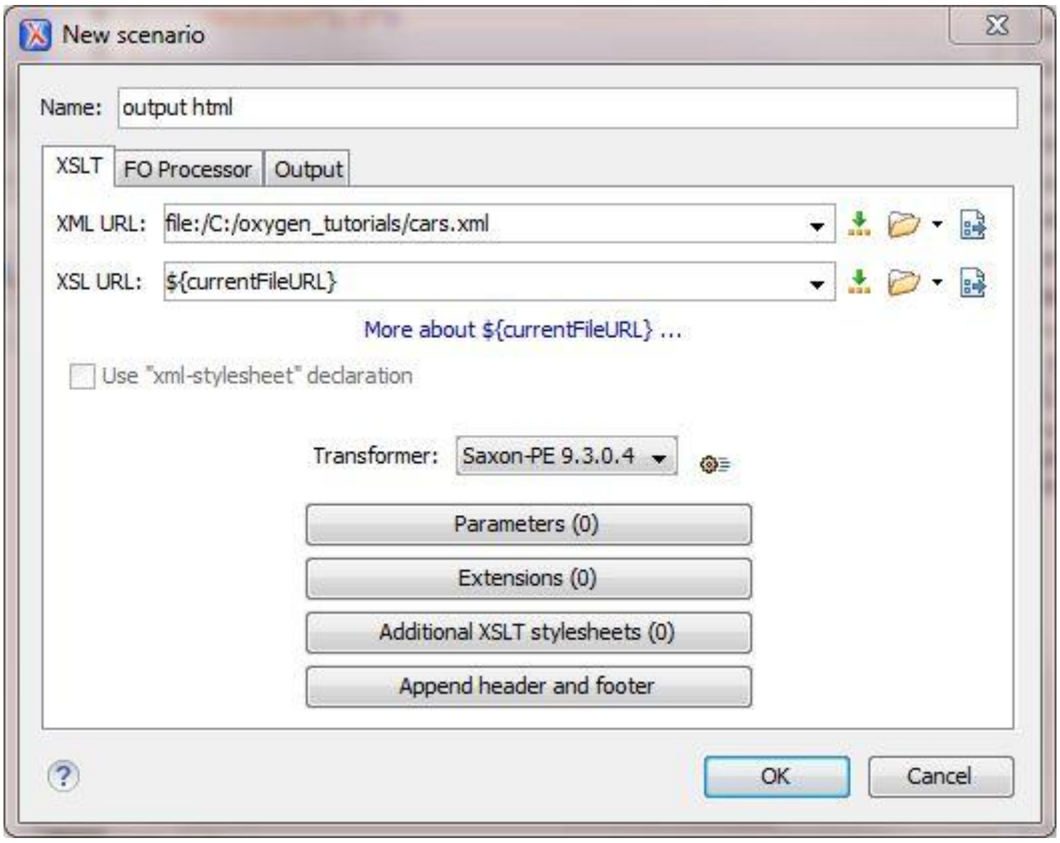
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs" version="2.0">
  <xsl:output doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN" indent="yes"/>
  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head><title></title></head>
      <body>
        <table border="0">
          <tr>
            <th>Maker</th>
            <th>Model</th>
          </tr>
          <xsl:for-each select="//car">
            <xsl:sort select="maker"/>
            <xsl:sort select="model"/>
            <tr>
              <th><xsl:value-of select="maker"/></th>
              <th><xsl:value-of select="model"/></th>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

3.1 Configure XSL Transformation Scenario

Now we want to run this XSL transformation. First we need to configure a transformation scenario. Press the button ”Configure Transformation Scenario. A dialog that is very similar to the one used to configure XQuery transformation scenario appears, but here we have to do a little bit more configuration. Press ”New”.



A tabbed dialog named ”New scenario” appears where you can change several things. In this case, simply point out the XML URL (the location of your cars.xml) for this transformation. There are several transformers to choose from, but in this course you will be using Saxon-B version 9.1.0.7 for your XSL transformations, this is the default transformer. Press OK to dismiss the dialog when you are satisfied with the settings (you can ignore the other tabs in the dialog for now).



New scenario

Name: output html

XML URL: file:/C:/oxygen_tutorials/cars.xml

XSL URL: ${currentFileURL}

Use "xml-stylesheet" declaration

Transformer: Saxon-PE 9.3.0.4

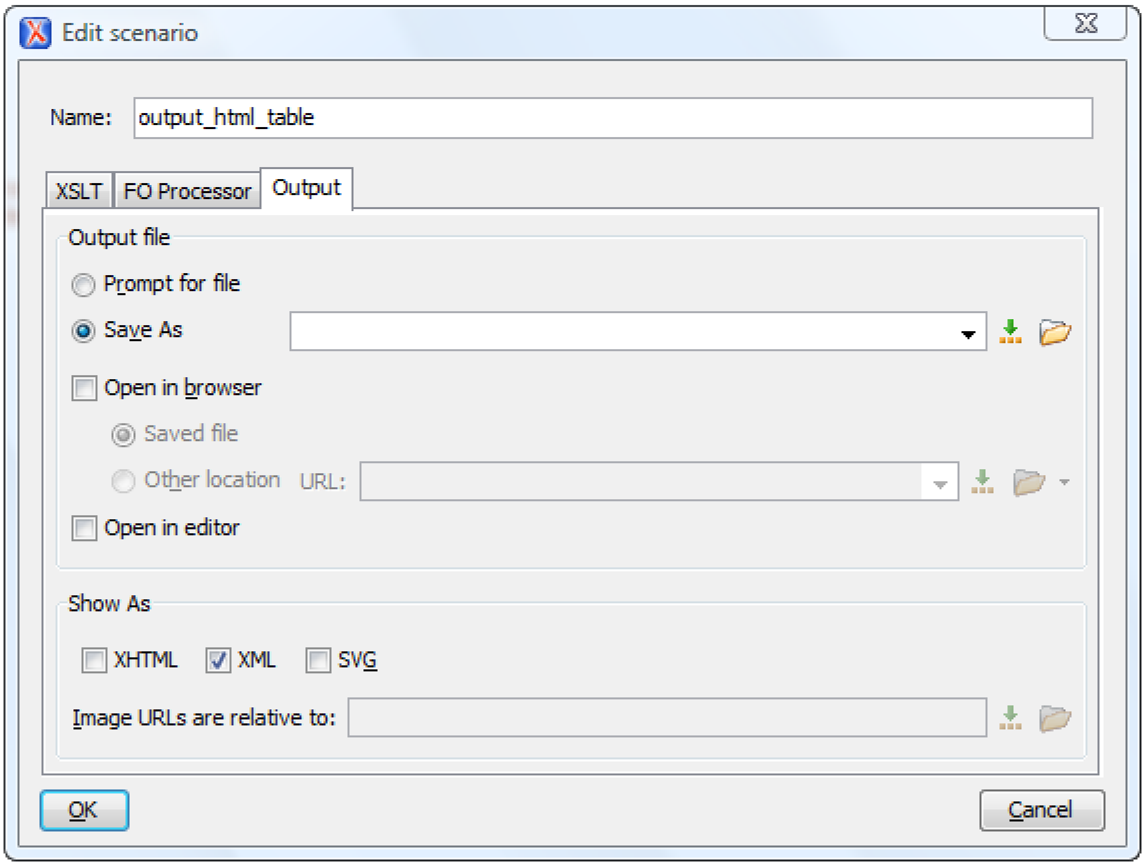
3.2 Execute Transformation

Now we are ready to perform the XSL transformation and we do that the same way we executed the XQuery transformation: by clicking the button ”Apply Transformation Scenario”. If you followed the instructions carefully the following result of the XSL transformation should appear in the bottom part of the oXygen window:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <title></title>
   </head>
   <body>
      <table border="0">
         <tr>
            <th>Maker</th>
            <th>Model</th>
         </tr>
         <tr>
            <th>BMW</th>
            <th>750Li</th>
         </tr>
         <tr>
            <th>BMW</th>
            <th>M5</th>
         </tr>
         <tr>
            <th>Ford</th>
            <th>F150</th>
         </tr>
         <tr>
            <th>Volvo</th>
            <th>V70</th>
         </tr>
      </table>
   </body>
</html>
```

3.3 Configure XSL Output

Raw text dumps like that are not the only way of outputting the result is not the only way, however. One could select the output to be saved to a name file and/or opened in a browser, for instance. Output options are found on the ”Output”-tab on the ”Edit scenario”-dialog you just used. If you want to change the output options of an existing transformation scenario, simply click ”Configure Transformation Scenario”, select the one you are interested in and press the ”Edit” button, the select the ”Output”-tab. Here's what it looks like:



Name: output_html_table

Output file

Prompt for file [✓ Save As]

Open in browser [ ]

[✓ Saved file] Other location URL:

Open in editor [ ]

Show As

XHTML [ ] [✓ XML] SVG [ ]

Image URLs are relative to:

4. A few tips...

If you want you can create so called projects in oXygen where you can group related files. You don't have to work using oXygen projects in the course but some might find it easier to organize things that way.

To toggle comments on the current selection: Ctrl + Shift + ,

That key combination is very useful indeed!

Most of the time oXygen indents for you automatically, but certain times you have to indent blocks of text manually. The key combination for that is: Ctrl + i

However, sometimes the indentation mechanism refuses to do its work when Ctrl + i is pressed and then selecting the same thing from the menu usually works. That menu item can be found at ”Document->Source->Indent Selection”.

When you output HTML using XSL transformations, don't forget to validate the result.

This was just a very short introduction of how oXygen works to get you started on the labs, explore the program yourselves!