Delphi® XE2
The fastest way to build native
Windows, Mac and iOS applications
Windows
Mac
iOS
Cloud
64-bit
FireMonkey
Database

XSL transforming XML

Jeroen Pluimers
jpluimers@better-office.com
better office benelux

DelphiLive 2011
DE, Düsseldorf, 20111026

Agenda

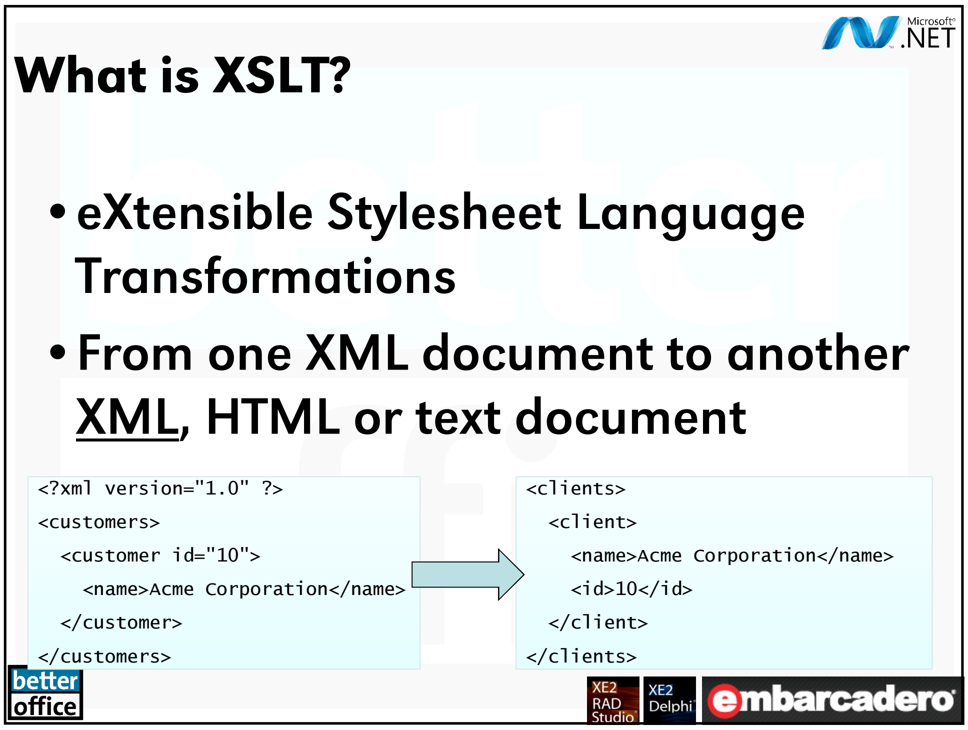
- What is XSLT?
- XPath
- XSLT

What is XSLT?

What is XSLT?

- eXtensible Stylesheet Language Transformations
- From one XML document to another XML, HTML or text document

```
<?xml version="1.0" ?>
<customers>
  <customer id="10">
    <name>Acme Corporation</name>
  </customer>
</customers>
```

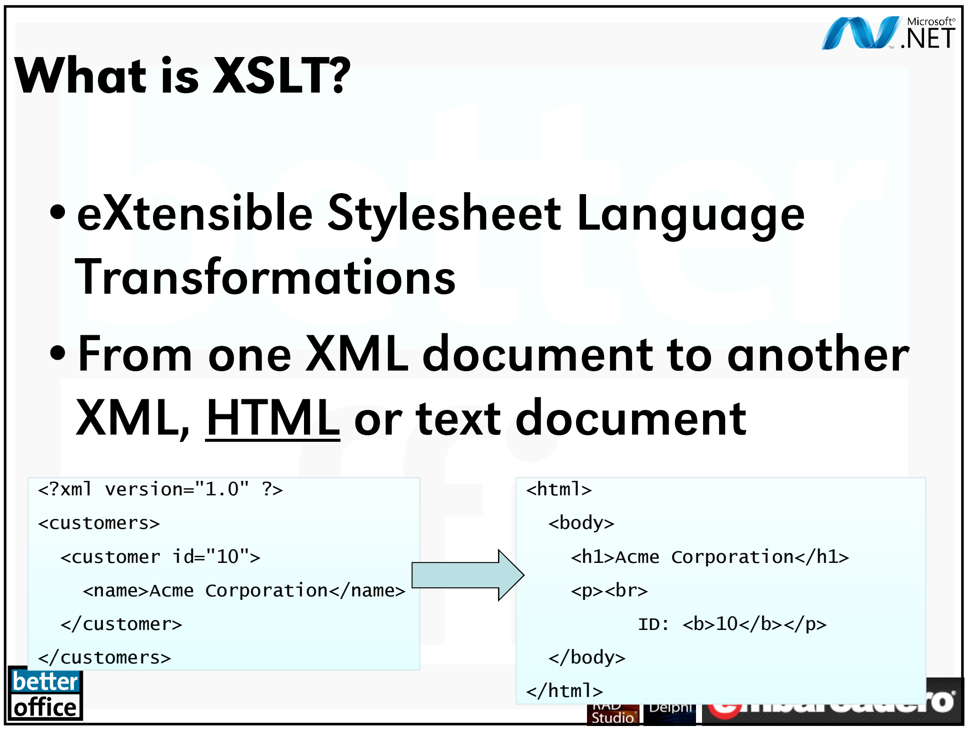


What is XSLT?

- eXtensible Stylesheet Language Transformations
- From one XML document to another XML, HTML or text document

```
<?xml version="1.0" ?>
<customers>
  <customer id="10">
    <name>Acme Corporation</name>
  </customer>
</customers>
```

```
<clients>
  <client>
    <name>Acme Corporation</name>
    <id>10</id>
  </client>
</clients>
```



What is XSLT?

- eXtensible Stylesheet Language Transformations
- From one XML document to another XML, HTML or text document

```
<?xml version="1.0" ?>
<customers>
  <customer id="10">
    <name>Acme Corporation</name>
  </customer>
</customers>
```

```
<html>
  <body>
    <h1>Acme Corporation</h1>
    <p><br>
          ID: <b>10</b></p>
  </body>
</html>
```

What is XSLT?

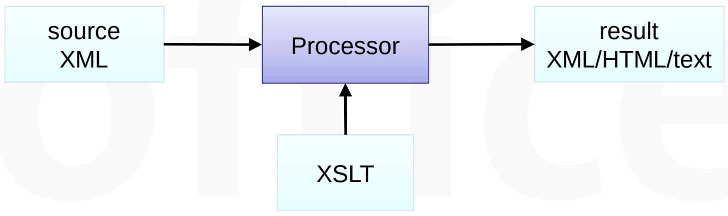
- eXtensible Stylesheet Language Transformations
- From one XML document to another XML, HTML or <u>text</u> document

```
<?xml version="1.0" ?>
<customers>
  <customer id="10">
    <name>Acme Corporation</name>
  </customer>
</customers>
```

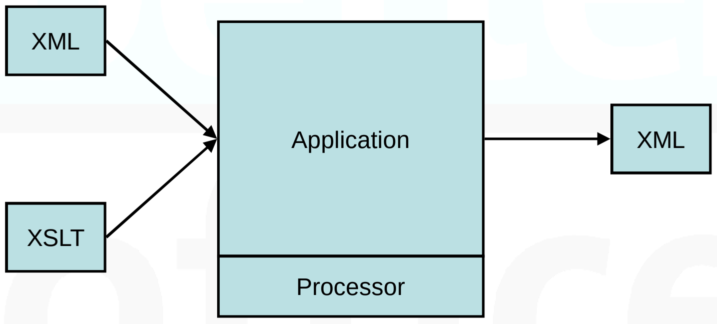
```
Customer "Acme Corporation"
at index 10
```

XSLT Document

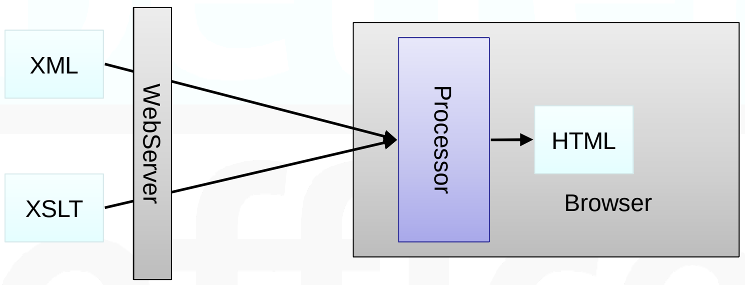
- <u>Describes</u> how a *source document* will needs be transformed into a *result document*



XSLT Scenarios



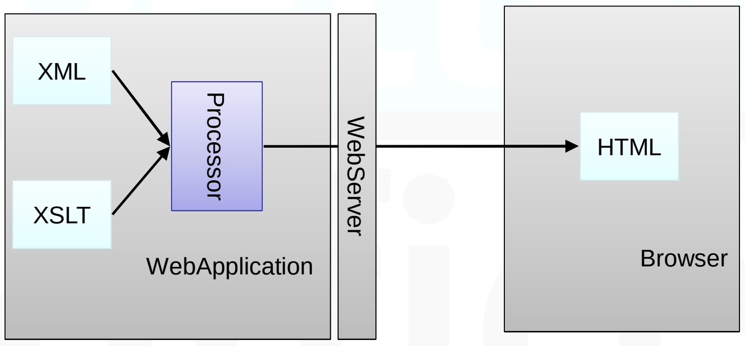
XSLT Scenarios



res://msxml.dll/defaultss.xsl

res://msxml3.dll/DEFAULTSS.xsl

XSLT Scenarios



Windows XSLT console processors

- nxslt
  - .NET based
  - Outputs Unicode UTF8
  - Download used to be here
    - http://xmlwriter.net/user_tools/nxslt.shtml
- msxsl
  - MS XML based
  - Outputs Unicode UTF16
  - Download:
    - http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=21714
- Batch file to convert

```
@nxslt %* | utf82ascii
@msxsl %* | utf162utf8
```

  - Download:
    - http://bo.codeplex.com/SourceControl/changeset/view/70711#1489616
  - utf82ascii and utf162ascii are also at http://bo.codeplex.com in both Delphi and C# varieties

XPath

XPath: localising data

- The first “/” indicates the *root* of the *source-document.*
  - /customers/customer/name
    ① ② ③

```
 <?xml version ="1.0" ?>
①<customers>
 ②<customer id="10">
   ③<name>Acme Corporation</name>
   </customer>
 </customers>
```

XPath: localising data

- Double "//" localises all nodes with a name, independent of the location inside the source-document:
  - //customer ①

- Attributes are indicated by a @
  - /customers/customer/@id ②

```
<?xml version ="1.0" ?>
<customers>
① <customer id="10">
   <name>Acme Corporation</name>
 </customer>
</customers>
```

```
<?xml version ="1.0" ?>
<customers>   ②
 <customer id="10">
   <name>Acme Corporation</name>
 </customer>
</customers>
```

Simplified stylesheets

- Can only have one template
- Create an HTML document
- Add to the root-element:

  -
    ```
    xsl:version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    ```
- Use <xsl:value-of> to obtain values from the source-document:

  -
    ```
    <xsl:value-of
           select="/customers/customer/name" />
    ```

Simplified stylesheet demo

```
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <head>
          <title>Customer Simplified StyleSheet demo</title>
    </head>
    <body>
     <h1><xsl:value-of select="/customers/customer/name" /></h1>
     <p>                          1
     <br/>           !!
     ID: <b><xsl:value-of select="/customers/customer/@id" /></b>
     </p>                            2
    </body>
</html>
```

```
<?xml version="1.0" ?>
<customers>
  <customer id="10">2
1 <name>Acme Corporation</name>
  </customer>
</customers>
```

```
<html>
  <body> 1
    <h1>Acme Corporation</h1>
    <p><br> !!   2
          ID:  <b>10</b></p>
  </body>
</html>
```

better office

Composite stylesheets

- More possibilities
- More flexibility
- Based on XPath templates
- XSLT document start with:

  `<xsl:stylesheet>`
- or with:

  `<xsl:transform>`

better office

XE2 RAD Studio XE2 Delphi embarcadero

Composite stylesheet demo

```
<xsl:stylesheet version="1.0“ mlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Customer Simplified StyleSheet demo</title>
      </head>
      <body>
        <h1>
          <xsl:value-of select="/customers/customer/name"/>
        </h1>
        <p>
        <br/>
     ID: <b>
            <xsl:value-of select="/customers/customer/@id"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
   <name>Acme Corporation</name>
 </customer>
</customers>
```

```
<html>
  <body>
    <h1>Acme Corporation</h1>
    <p><br>
          ID: <b>10</b></p>
  </body>
html>
```

better office

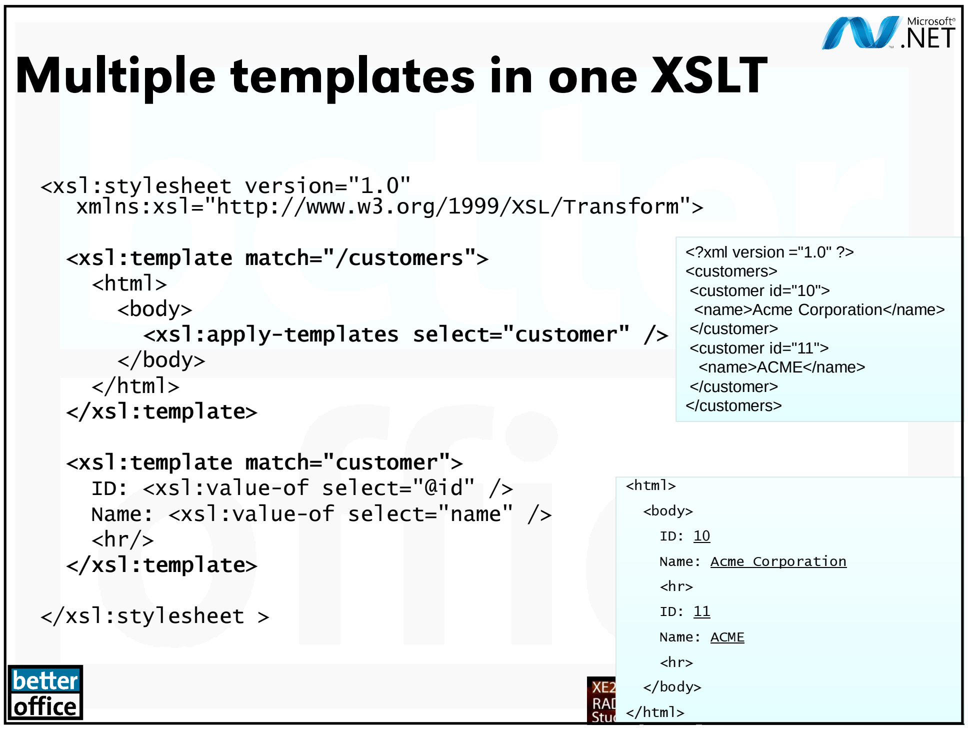
Composite stylesheet demo

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Customer Simplified StyleSheet demo</title>
      </head>
      <body>
        <h1>
          <xsl:value-of select="/customers/customer/name"/>
        </h1>
        <p>
        <br/>
     ID: <b>
            <xsl:value-of select="/customers/customer/@id"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
   <name>Acme Corporation</name>
 </customer>
</customers>
```

```
<html>
  <body>
    <h1>Acme Corporation</h1>
    <p><br>
          ID: <b>10</b></p>
  </body>
html>
```

better office

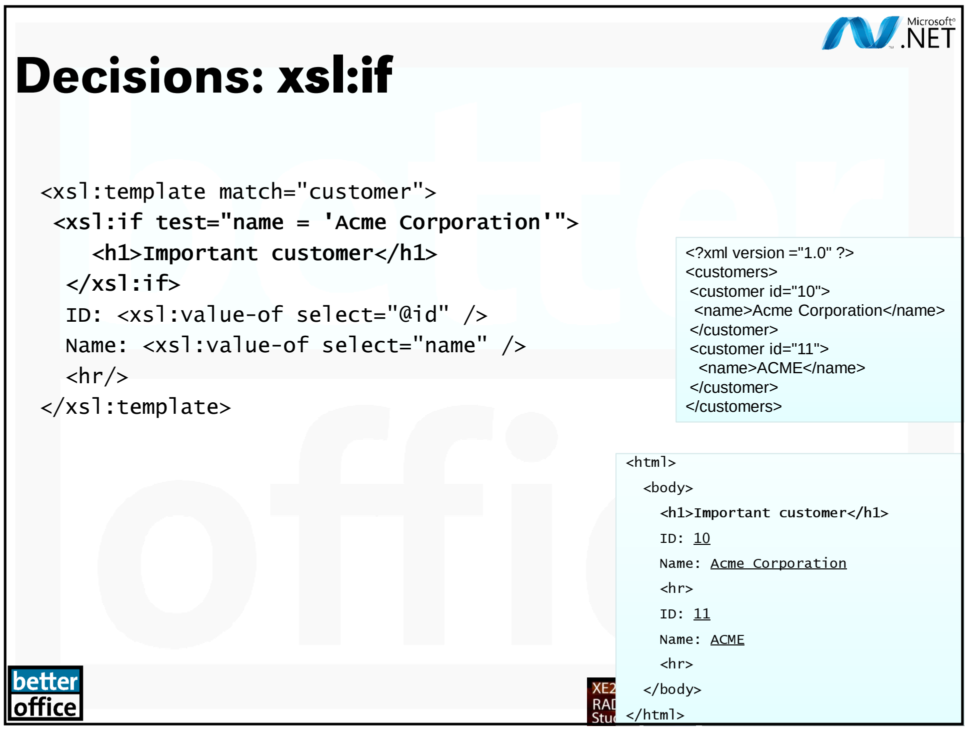


Multiple templates in one XSLT

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/customers">
    <html>
      <body>
        <xsl:apply-templates select="customer" />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="customer">
    ID: <xsl:value-of select="@id" />
    Name: <xsl:value-of select="name" />
    <hr/>
  </xsl:template>

</xsl:stylesheet >
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
  <name>Acme Corporation</name>
 </customer>
 <customer id="11">
   <name>ACME</name>
 </customer>
</customers>
```

```
<html>
  <body>
    ID: 10
    Name: Acme Corporation
    <hr>
    ID: 11
    Name: ACME
    <hr>
  </body>
</html>
```

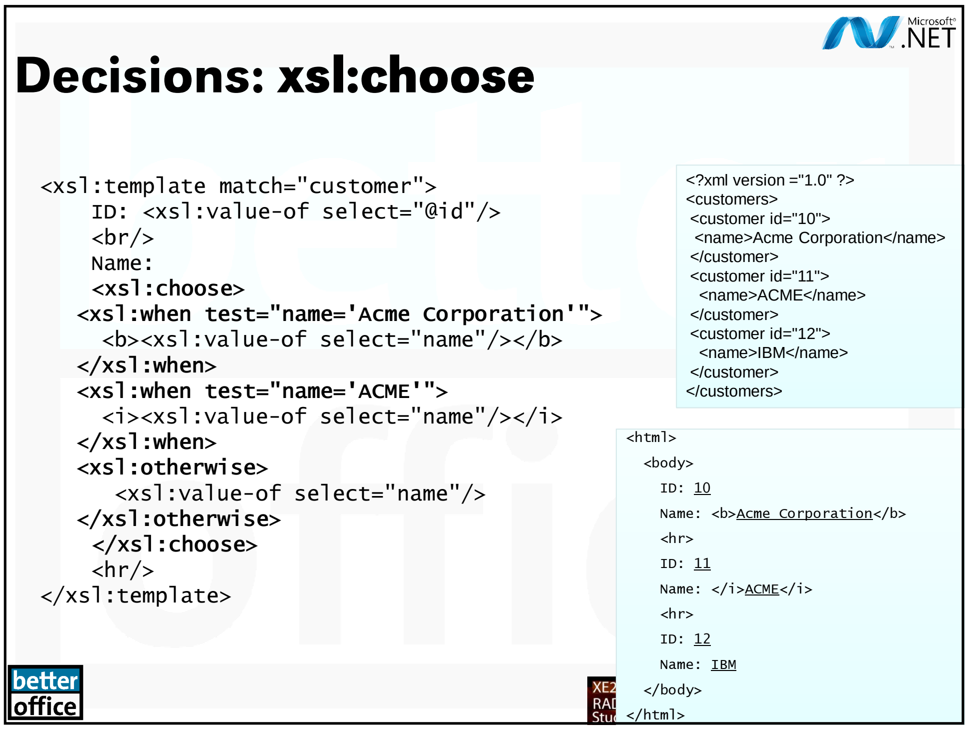


Decisions: xsl:if

```
<xsl:template match="customer">
 <xsl:if test="name = 'Acme Corporation'">
    <h1>Important customer</h1>
  </xsl:if>
  ID: <xsl:value-of select="@id" />
  Name: <xsl:value-of select="name" />
  <hr/>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
  <name>Acme Corporation</name>
 </customer>
 <customer id="11">
   <name>ACME</name>
 </customer>
</customers>
```

```
<html>
  <body>
    <h1>Important customer</h1>
    ID: 10
    Name: Acme Corporation
    <hr>
    ID: 11
    Name: ACME
    <hr>
  </body>
</html>
```



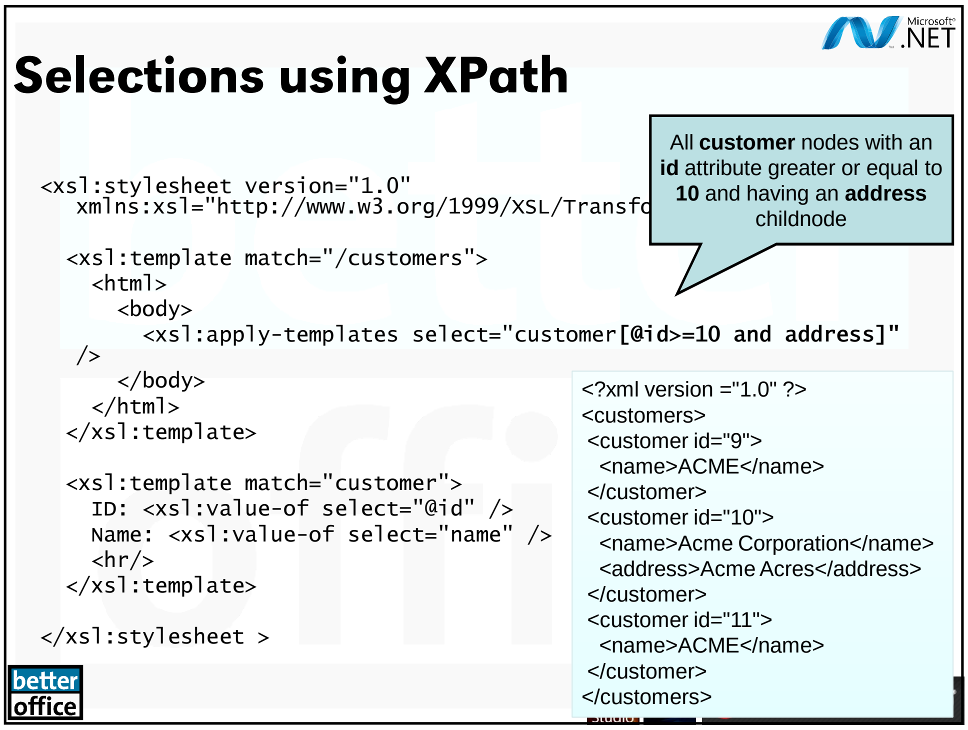
Decisions: xsl:choose

```
<xsl:template match="customer">
    ID: <xsl:value-of select="@id"/>
    <br/>
    Name:
    <xsl:choose>
  <xsl:when test="name='Acme Corporation'">
    <b><xsl:value-of select="name"/></b>
  </xsl:when>
  <xsl:when test="name='ACME'">
    <i><xsl:value-of select="name"/></i>
  </xsl:when>
  <xsl:otherwise>
     <xsl:value-of select="name"/>
  </xsl:otherwise>
    </xsl:choose>
    <hr/>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
  <name>Acme Corporation</name>
 </customer>
 <customer id="11">
   <name>ACME</name>
 </customer>
 <customer id="12">
   <name>IBM</name>
 </customer>
</customers>
```

```
<html>
  <body>
    ID: 10
    Name: <b>Acme Corporation</b>
    <hr>
    ID: 11
    Name: </i>ACME</i>
    <hr>
    ID: 12
    Name: IBM
  </body>
</html>
```

More XPath



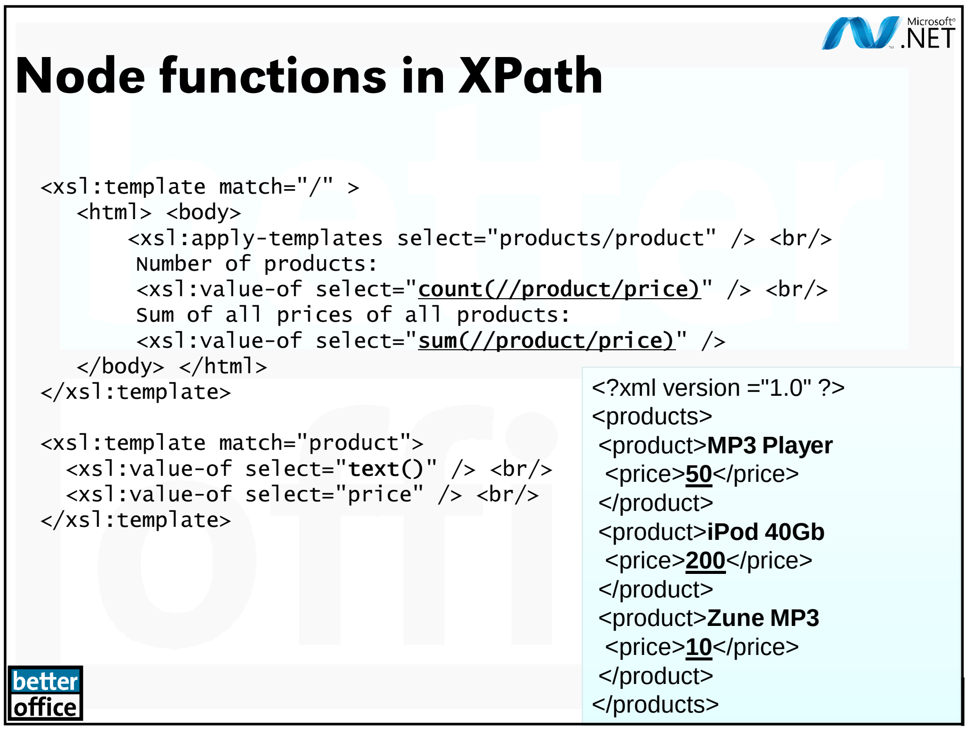
Selections using XPath

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transfo

  <xsl:template match="/customers">
    <html>
      <body>
        <xsl:apply-templates select="customer[@id>=10 and address]"
   />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="customer">
    ID: <xsl:value-of select="@id" />
    Name: <xsl:value-of select="name" />
    <hr/>
  </xsl:template>

</xsl:stylesheet >
```

All **customer** nodes with an **id** attribute greater or equal to **10** and having an **address** childnode

```
<?xml version ="1.0" ?>
<customers>
 <customer id="9">
   <name>ACME</name>
 </customer>
 <customer id="10">
   <name>Acme Corporation</name>
   <address>Acme Acres</address>
 </customer>
 <customer id="11">
   <name>ACME</name>
 </customer>
</customers>
```

better office



Node functions in XPath

```
<xsl:template match="/" >
   <html> <body>
       <xsl:apply-templates select="products/product" /> <br/>
        Number of products:
        <xsl:value-of select="count(//product/price)" /> <br/>
        Sum of all prices of all products:
        <xsl:value-of select="sum(//product/price)" />
   </body> </html>
</xsl:template>

<xsl:template match="product">
  <xsl:value-of select="text()" /> <br/>
  <xsl:value-of select="price" /> <br/>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<products>
 <product>MP3 Player
  <price>50</price>
 </product>
 <product>iPod 40Gb
  <price>200</price>
 </product>
 <product>Zune MP3
  <price>10</price>
 </product>
</products>
```

better office

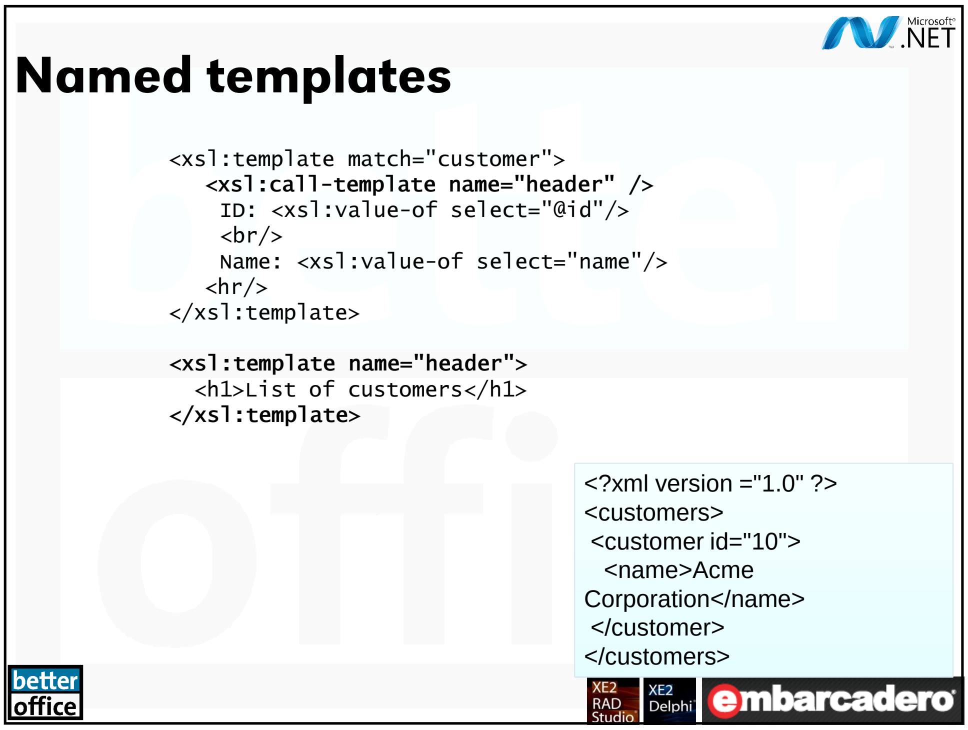
Node set functions in XPath

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/customers">
    <html>
      <body>
        <xsl:apply-templates select="customer[position() = last()]"
   />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="customer">
    ID: <xsl:value-of select="@id" />
    Name: <xsl:value-of select="name" />
    <hr/>
  </xsl:template>

</xsl:stylesheet >
```

The final customer node

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
  <name>Acme Corporation</name>
 </customer>
 <customer id="11">
   <name>ACME</name>
 </customer>
 <customer id="12">
   <name>IBM</name>
 </customer>
</customers>
```

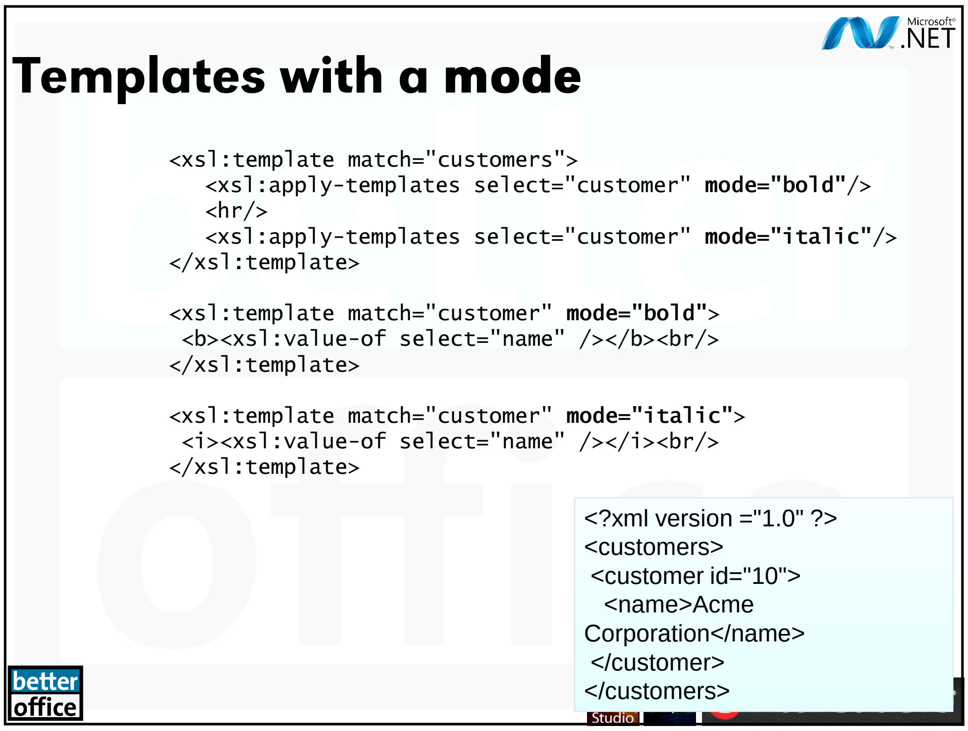
Back to XSLT



Named templates

```
<xsl:template match="customer">
   <xsl:call-template name="header" />
    ID: <xsl:value-of select="@id"/>
    <br/>
    Name: <xsl:value-of select="name"/>
   <hr/>
</xsl:template>

<xsl:template name="header">
  <h1>List of customers</h1>
</xsl:template>
```

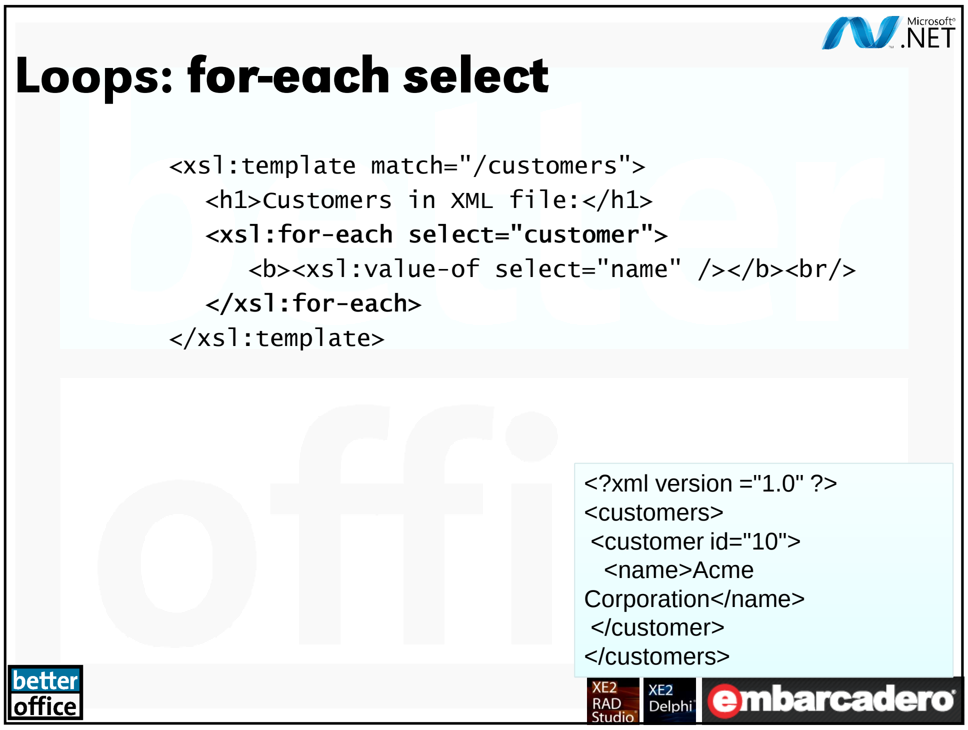
```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
   <name>Acme
Corporation</name>
 </customer>
</customers>
```



Templates with a mode

```
<xsl:template match="customers">
   <xsl:apply-templates select="customer" mode="bold"/>
   <hr/>
   <xsl:apply-templates select="customer" mode="italic"/>
</xsl:template>

<xsl:template match="customer" mode="bold">
 <b><xsl:value-of select="name" /></b><br/>
</xsl:template>

<xsl:template match="customer" mode="italic">
 <i><xsl:value-of select="name" /></i><br/>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
   <name>Acme
Corporation</name>
 </customer>
</customers>
```



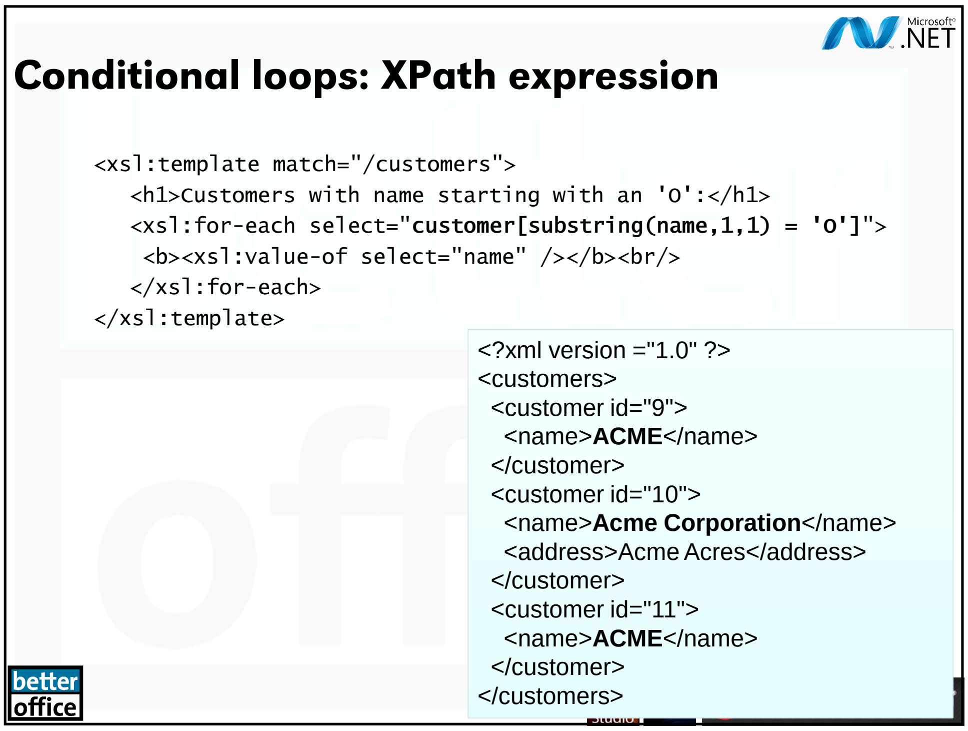
Loops: for-each select

```
<xsl:template match="/customers">
   <h1>Customers in XML file:</h1>
   <xsl:for-each select="customer">
      <b><xsl:value-of select="name" /></b><br/>
   </xsl:for-each>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
 <customer id="10">
   <name>Acme
Corporation</name>
 </customer>
</customers>
```

better office

XE2 RAD Studio XE2 Delphi embarcadero

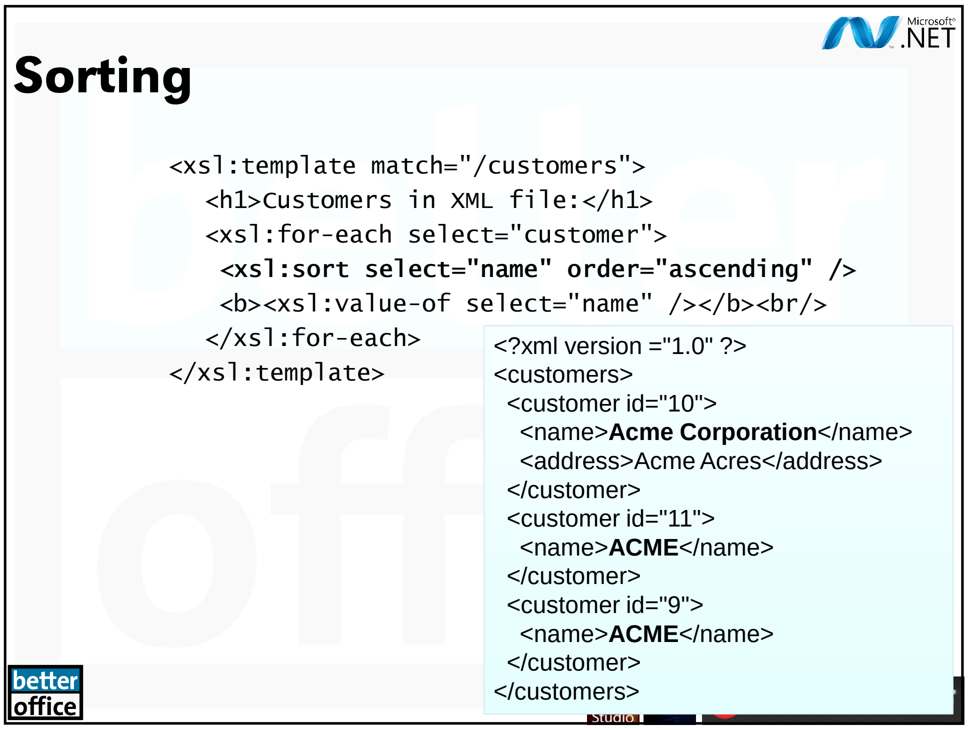


Conditional loops: XPath expression

```
<xsl:template match="/customers">
   <h1>Customers with name starting with an 'O':</h1>
   <xsl:for-each select="customer[substring(name,1,1) = 'O']">
    <b><xsl:value-of select="name" /></b><br/>
   </xsl:for-each>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
  <customer id="9">
    <name>ACME</name>
  </customer>
  <customer id="10">
    <name>Acme Corporation</name>
    <address>Acme Acres</address>
  </customer>
  <customer id="11">
    <name>ACME</name>
  </customer>
</customers>
```

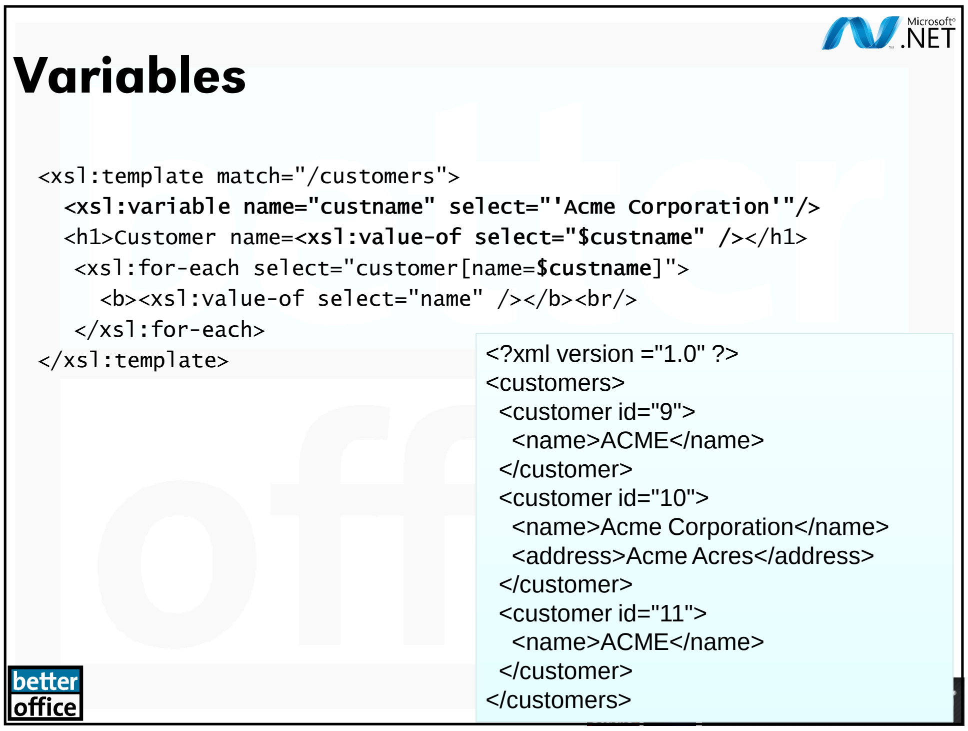
better office



Sorting

```
<xsl:template match="/customers">
   <h1>Customers in XML file:</h1>
   <xsl:for-each select="customer">
    <xsl:sort select="name" order="ascending" />
    <b><xsl:value-of select="name" /></b><br/>
   </xsl:for-each>
</xsl:template>
```

```
<?xml version ="1.0" ?>
<customers>
  <customer id="10">
    <name>Acme Corporation</name>
    <address>Acme Acres</address>
  </customer>
  <customer id="11">
    <name>ACME</name>
  </customer>
  <customer id="9">
    <name>ACME</name>
  </customer>
</customers>
```



Variables

```
<xsl:template match="/customers">
  <xsl:variable name="custname" select="'Acme Corporation'"/>
  <h1>Customer name=<xsl:value-of select="$custname" /></h1>
   <xsl:for-each select="customer[name=$custname]">
     <b><xsl:value-of select="name" /></b><br/>
   </xsl:for-each>
</xsl:template>
```

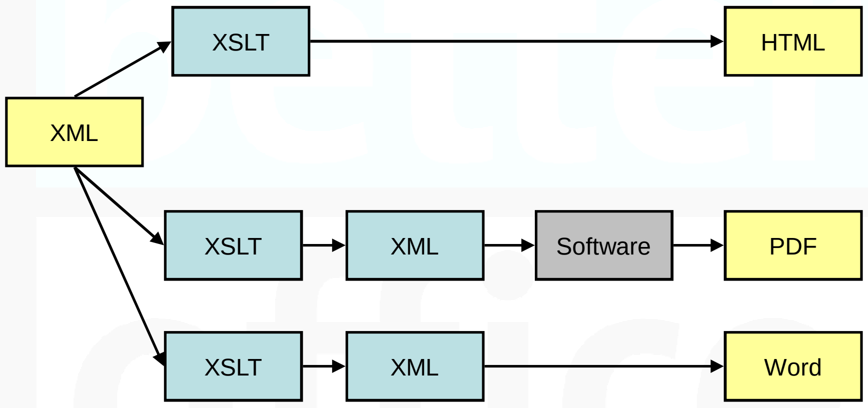
```
<?xml version ="1.0" ?>
<customers>
  <customer id="9">
    <name>ACME</name>
  </customer>
  <customer id="10">
    <name>Acme Corporation</name>
    <address>Acme Acres</address>
  </customer>
  <customer id="11">
    <name>ACME</name>
  </customer>
</customers>
```

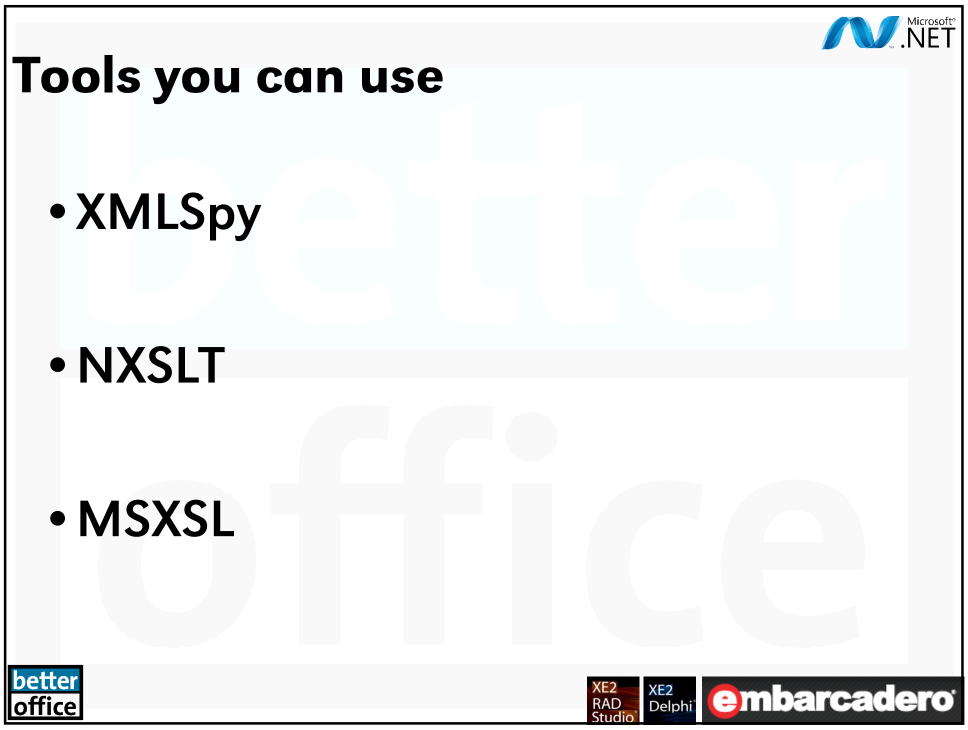
Refencing other XML

```
<xsl:param name="examxml" select="'ExamDescriptions.xml'"/>
 <xsl:variable name="lookupTable" select="document($examxml)"/>
 <xsl:template match="/">
   <xsl:for-each select="/Exams/Exam">
  <xsl:variable name="CurrentCode" select="."/>
    <xsl:variable name="CurrentExam"
      select="$lookupTable/Exams/Exam[ @Code = $CurrentCode ]"/>
    <xsl:value-of select="."/>
    <xsl:value-of select="$CurrentExam/Name"/>
    <br/>
  </xsl:for-each>
</xsl:template>
```

```
Note: select == default value of parameter.
```

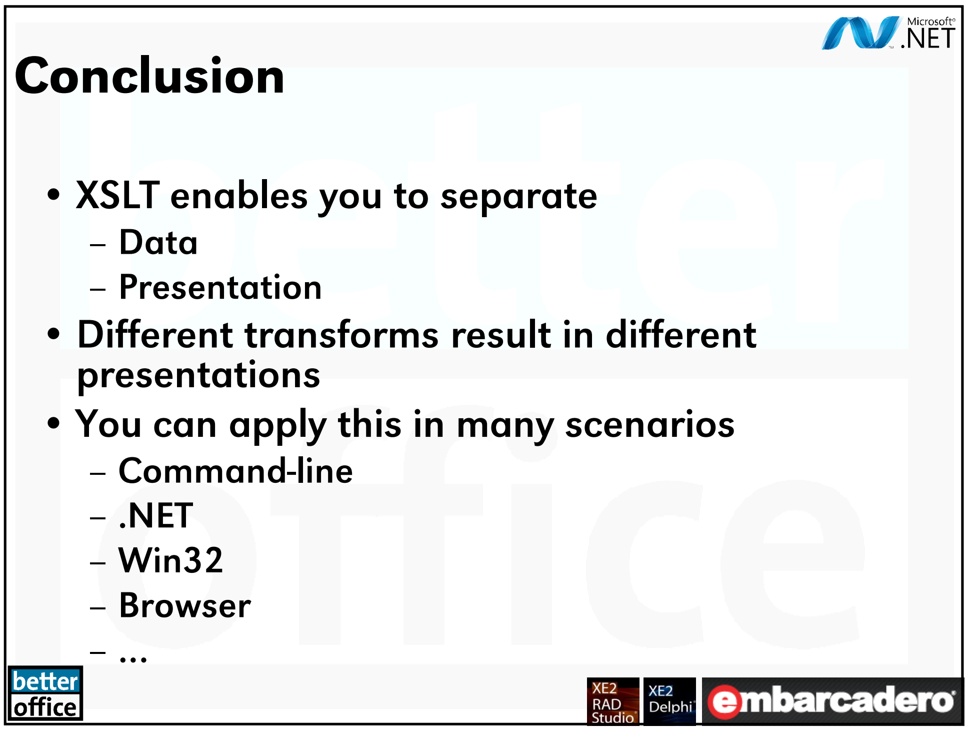
Possible: Presenting documents





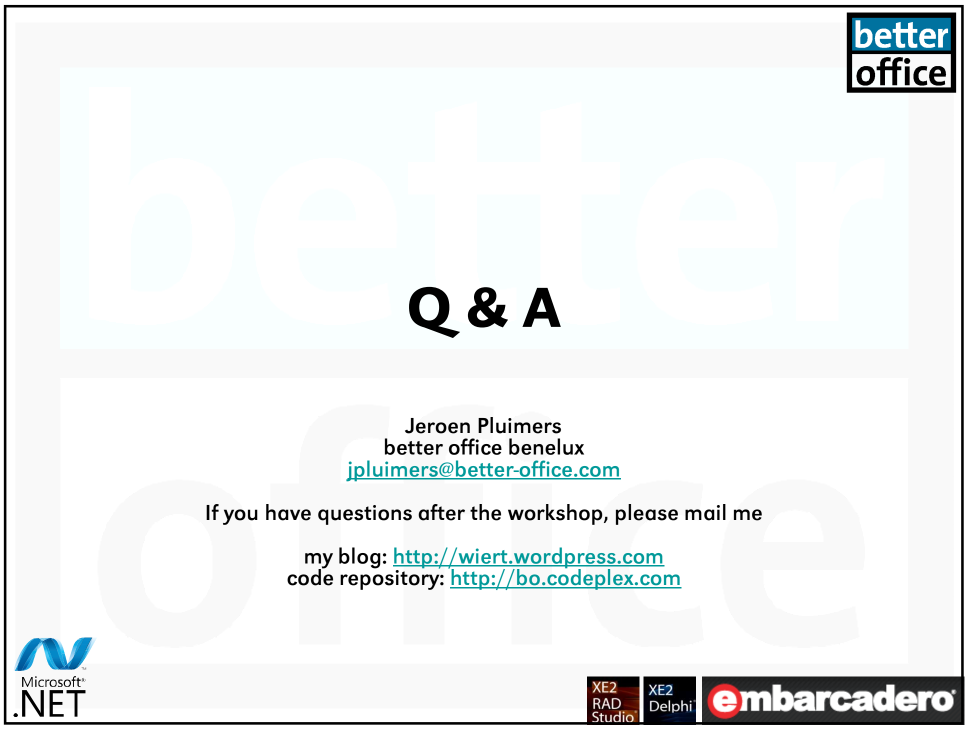
Tools you can use

- XMLSpy
- NXSLT
- MSXSL



Conclusion

- XSLT enables you to separate
  - Data
  - Presentation
- Different transforms result in different presentations
- You can apply this in many scenarios
  - Command-line
  - .NET
  - Win32
  - Browser
  - …



Q & A

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

If you have questions after the workshop, please mail me

my blog: http://wiert.wordpress.com
code repository: http://bo.codeplex.com

better office
Microsoft .NET
XE2 RAD Studio
XE2 Delphi
embarcadero