ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 7: XSLT - 2

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο v. 3.0

(Attribution – Non Commercial – Non-derivatives)

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML
XSLT Part 2
605.444 / 635.444

David Silberberg

Lecture 15

Conditional Processing

- <xsl:if test="Boolean expression">

or

- <xsl:choose>
  <xsl:when test="Boolean expression">
  <xsl:when test="Boolean expression">
  <xsl:otherwise>
  </xsl:choose>
- Boolean expression is XPath that is converted to a boolean

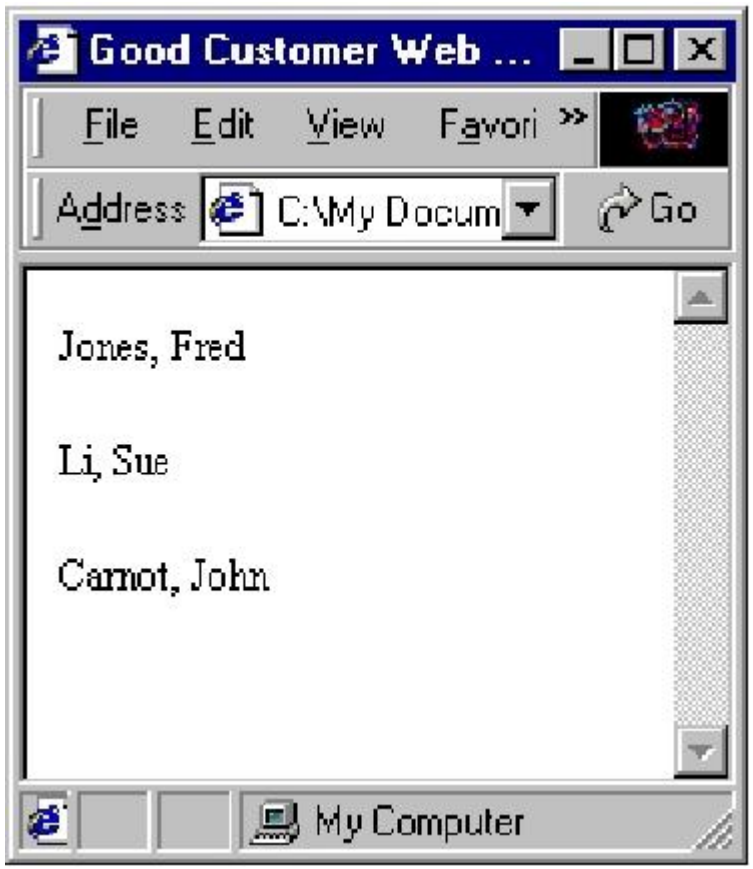
If Example

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
 version='1.0'>
<xsl:template match='/'>
    <html><head><title>Good Customer Web Page</title></head>
    <body><xsl:apply-templates/></body>
    </html>
</xsl:template>
<xsl:template match='//name'>
    <xsl:if test="name(..) = 'good'">
      <P><xsl:value-of select="."/></P>
    </xsl:if>
 </xsl:template>
</xsl:stylesheet>
```

If Output

```
<html>
<head>
<title>Good Customer Web Page</title>
</head>
<body>

    <P>Jones, Fred</P>
    <P>Li, Sue</P>
    <P>Carnot, John</P>




</body>
</html>
```

If Display


Good Customer Web ...
File Edit View Favori
Address C:\My Docum
Go
Jones, Fred
Li, Sue
Carnot, John
My Computer

Modified customer.xml

```
<?xml version="1.0"?>
<customers>
    <good>
        <name id="1">Jones, Fred</name>
        <name id="2">Li, Sue</name>
        <name id="3">Carnot, John</name>
    </good>
    <bad>
        <name id="4">Tell, William</name>
        <name id="5">Carr, Sam</name>
    </bad>
</customers>
```

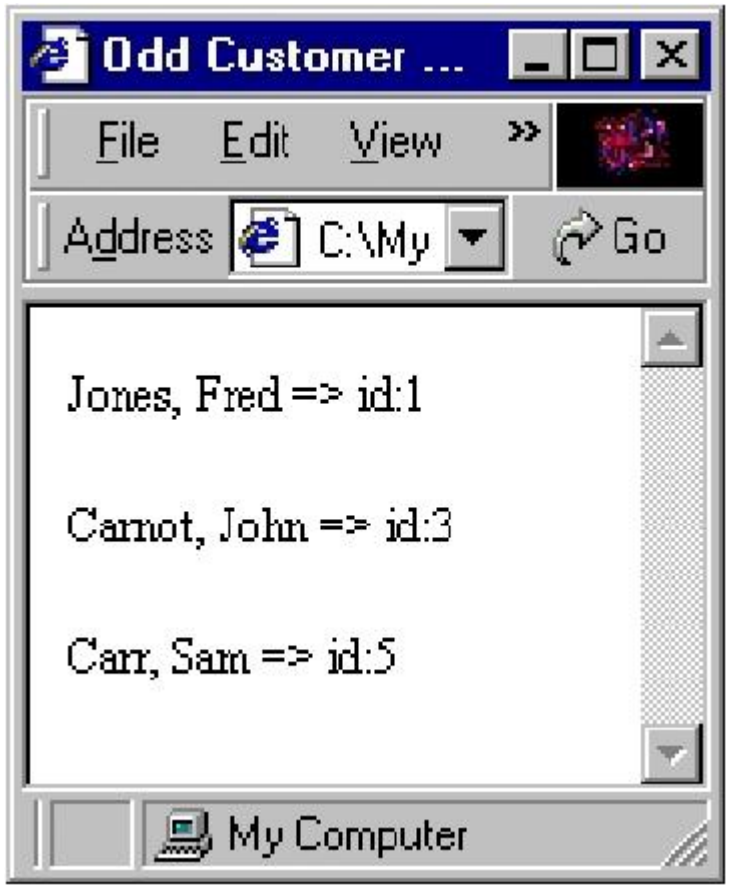
If Processing on Numeric Values

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
 version='1.0'>
<xsl:template match='/'>
   <html><head><title>Odd Customer Web Page</title></head>
   <body><xsl:apply-templates/></body></html>
</xsl:template>
<xsl:template match='//name'>
    <xsl:if test="number(./@id) mod 2 = 1">
      <P><xsl:value-of select="."/>
         <xsl:text> =&gt; id:</xsl:text>
         <xsl:value-of select="./@id"/></P>
   </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

If Output Results

```
<html>
<head>
<title>Odd Customer Web Page</title>
</head>
<body>

    <P>Jones, Fred =&gt; id:1</P>

    <P>Carnot, John =&gt; id:3</P>



    <P>Carr, Sam =&gt; id:5</P>

</body>
</html>
```

If Display


Odd Customer ...
File Edit View
Address C:\My
Go
Jones, Fred => id:1
Carnot, John => id:3
Carr, Sam => id:5
My Computer

Choose Example

- Provides the opportunity to provide several options on the output in one statement

```
<xsl:choose>
    <xsl:when test="number(./@id) &lt; 4">
       <P><xsl:value-of select="."/>
       <xsl:text> -- Good Guy!</xsl:text></P></xsl:when>
    <xsl:when test="number(./@id) &gt; 3">
       <P><xsl:value-of select="."/>
       <xsl:text> -- Bad Guy!</xsl:text></P></xsl:when>
    <xsl:otherwise>
       <P><xsl:value-of select="."/>
       <xsl:text> -- Sign Him Up!</xsl:text></P></xsl:otherwise>
</xsl:choose>
```

Choose Output

```
<html>
<head>
<title>Customer Web Page</title>
</head>
<body>

    <P>Jones, Fred -- Good Guy!</P>
    <P>Li, Sue -- Good Guy!</P>
    <P>Carnot, John -- Good Guy!</P>


    <P>Tell, William -- Bad Guy!</P>
    <P>Carr, Sam -- Bad Guy!</P>

</body>
</html>
```

Choose Display

Boolean Matches

- Match occurs when
  - Test value evaluates to true
  - If number, then must evaluate to something not equal to 0
- Otherwise occurs
  - If none of the tests evaluate to true
  - Could be value is text, not numeric
  - Could be that there is no node match

For Each

- <xsl:for-each select="XPath Expression">
- This is almost exactly like a template.
- <xsl:template …> cannot be embedded in another template
  - Templates must stand alone.
- <xsl:for-each …> can be embedded in a template.

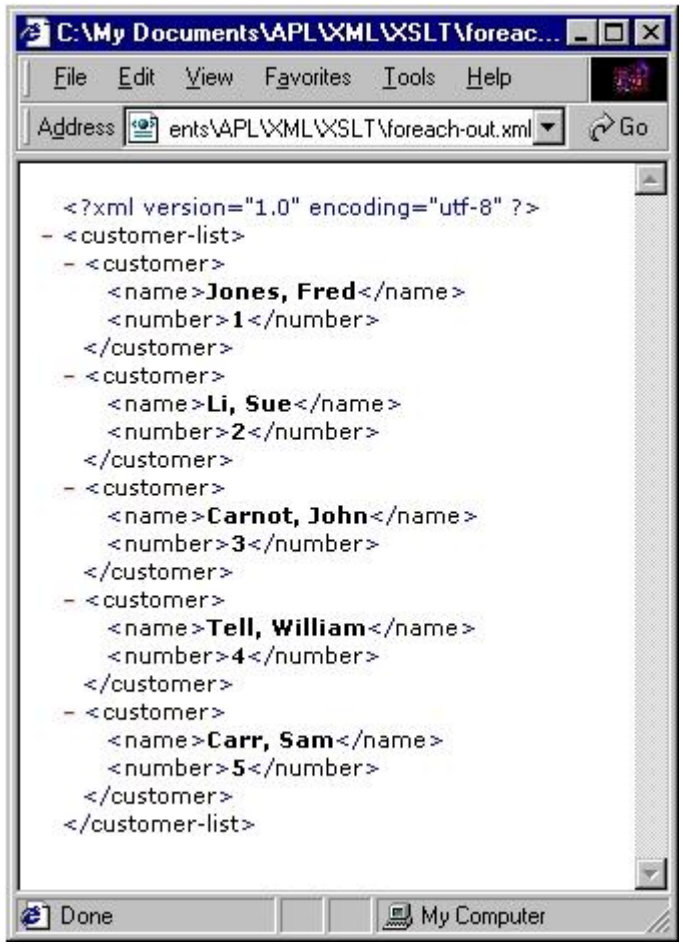
For Each Example - Without For-Each

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:template match='/'>
   <customer-list>
      <xsl:apply-templates/>
   </customer-list>
</xsl:template>
<xsl:template match='//name'>
   <customer>
   <name> <xsl:value-of select="."/> </name>
   <xsl:apply-templates select="@*"/>
   </customer>
</xsl:template>
<xsl:template match="@*">
   <number><xsl:value-of select="."/></number>
</xsl:template>
</xsl:stylesheet>
```

For Each Output - Without For-Each

```
<?xml version="1.0" encoding="utf-8"?>
<customer-list>

    <customer><name>Jones, Fred</name><number>1</number></customer>
    <customer><name>Li, Sue</name><number>2</number></customer>
    <customer><name>Carnot, John</name><number>3</number></customer>


    <customer><name>Tell, William</name><number>4</number></customer>
    <customer><name>Carr, Sam</name><number>5</number></customer>

</customer-list>
```

Without For Each Display



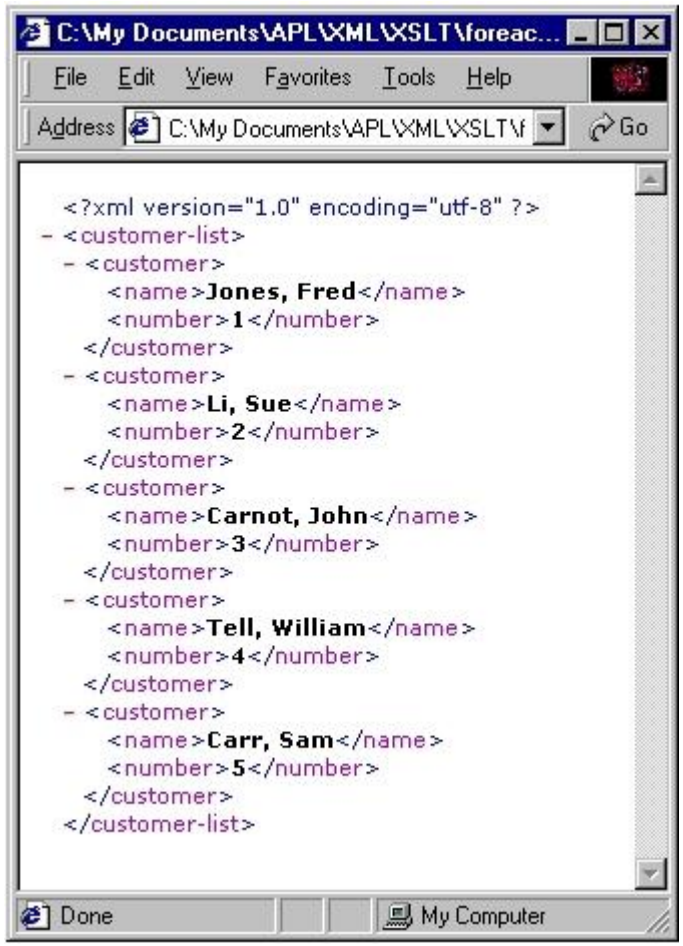
For Each Replacement

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:template match='/'>
   <customer-list>
      <xsl:apply-templates/>
   </customer-list>
</xsl:template>
<xsl:template match='//name'>
   <customer>
   <name> <xsl:value-of select="."/> </name>
   <xsl:for-each select="@*">
      <number><xsl:value-of select="."/></number>
   </xsl:for-each>
   </customer>
</xsl:template>
</xsl:stylesheet>
```

For Each Output

```
<?xml version="1.0" encoding="utf-8"?>
<customer-list>

    <customer><name>Jones, Fred</name><number>1</number></customer>
    <customer><name>Li, Sue</name><number>2</number></customer>
    <customer><name>Carnot, John</name><number>3</number></customer>


    <customer><name>Tell, William</name><number>4</number></customer>
    <customer><name>Carr, Sam</name><number>5</number></customer>

</customer-list>
```

For Each Display



Copy Of

- <xsl:copy-of select="XPath Expression"/>
- Provides capability of copying large sections of the source tree.
- This is convenient if the output is much like the input.
- It is more efficient than descending through the tree and executing <xsl:value-of …> on the element.

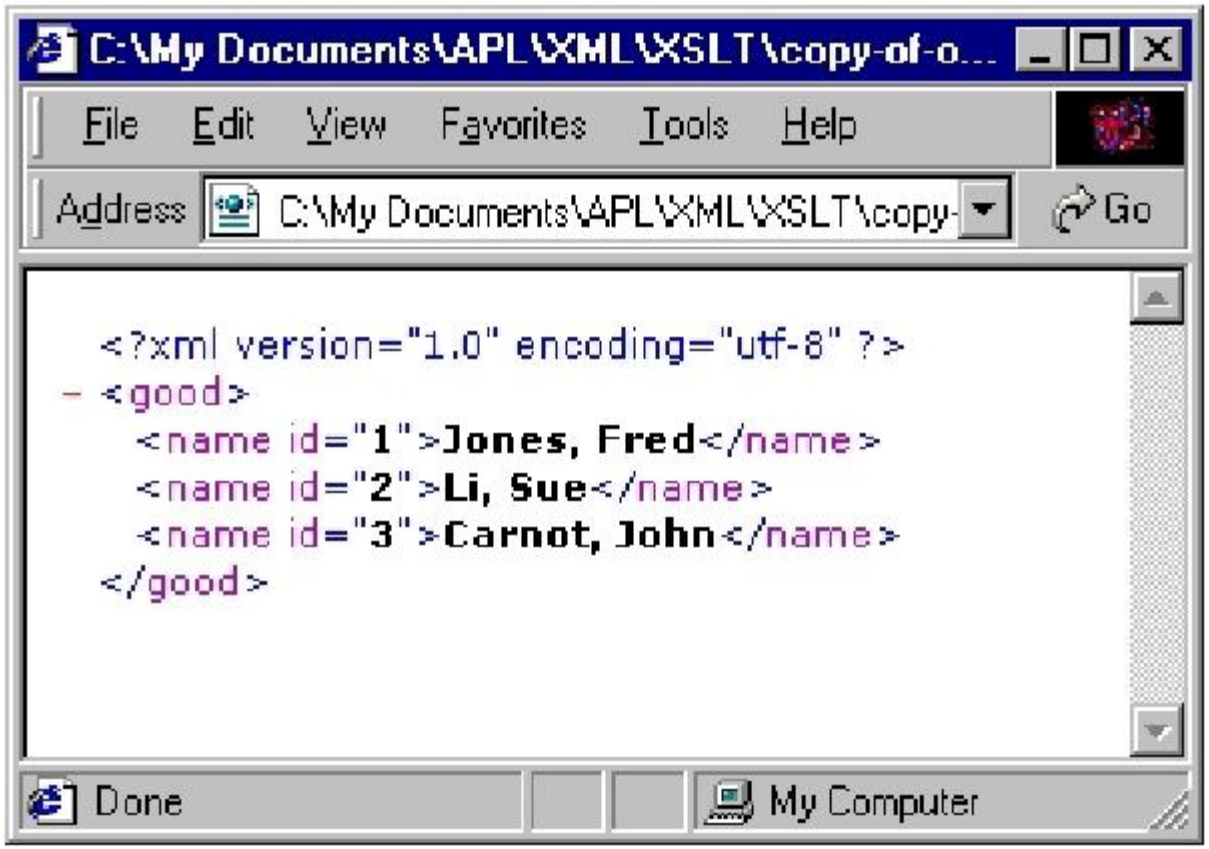
Copy Of Example

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
    version='1.0'>
<xsl:output indent="yes"/>

<xsl:template match='/'>
   <xsl:copy-of select="/customers/good"/>
</xsl:template>

</xsl:stylesheet>
```

Copy Of Output

```
<?xml version="1.0" encoding="utf-8"?>
<good>
        <name id="1">Jones, Fred</name>
        <name id="2">Li, Sue</name>
        <name id="3">Carnot, John</name>
    </good>
```

Copy Of Display



Copy

- <xsl:copy use-attribute-set=“attribute set names”>
- Copies current node, not the descendents
- However, if the current context is referenced in the body of the copy command, the descendents are referenced as well

Simple Copy Example & Output

- XSLT:

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:output indent="yes"/>
<xsl:template match='/customers'>
   <xsl:copy/>
</xsl:template>
</xsl:stylesheet>
```

- Yields:

```
<?xml version="1.0" encoding="utf-8"?>
<customers/>
```

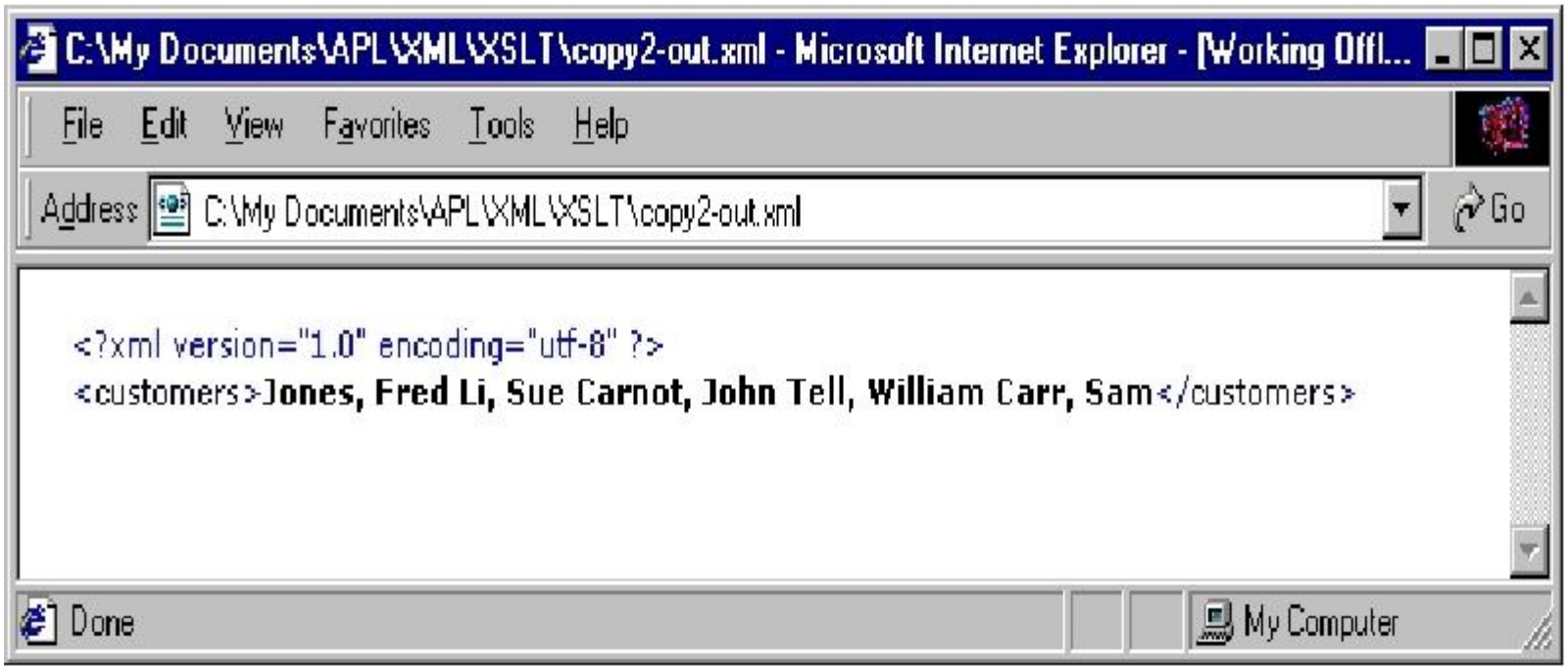
Copy with Context

```
<?xml version='1.0' ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
    version='1.0'>
<xsl:output indent="yes"/>

<xsl:template match='/customers'>
   <xsl:copy>
    <xsl:value-of select="."/>
   </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

Copy with Context Output

```
<?xml version="1.0" encoding="utf-8"?>
<customers>

    Jones, Fred
    Li, Sue
    Carnot, John



    Tell, William
    Carr, Sam

</customers>
```

Copy with Context Display



Sorting

- <xsl:sort select="XPath Expression"
  lang="lang"
  data-type="*text* or *number*"
  order="*ascending* or *descending*"
  case-order="*upper-first* or *lower-first*"/>
- Defaults are underlined
- select determines which element are to be sorted
  - Multiple xsl:sort commands will sort on the first command first
  - If sort values are identical, the output maintains its input order
- lang - the language of the text ("en", "fr", etc.)
- case-order – default is language dependent
- The rest is self explanatory.

New customer.xml Document

```
<?xml version="1.0"?>
<customers>
   <good>
      <name>
         <last>Jones</last>
         <first>Fred</first>
      </name>
      <name>
         <last>Li</last>
         <first>Sue</first>
      </name>
      <name>
         <last>Carnot</last>
         <first>John</first>
      </name>
   </good>
   <bad>
      <name>
         <last>Tell</last>
         <first>William</first>
      </name>
      <name>
         <last>Carr</last>
         <first>Sam</first>
      </name>
   </bad>
</customers>
```

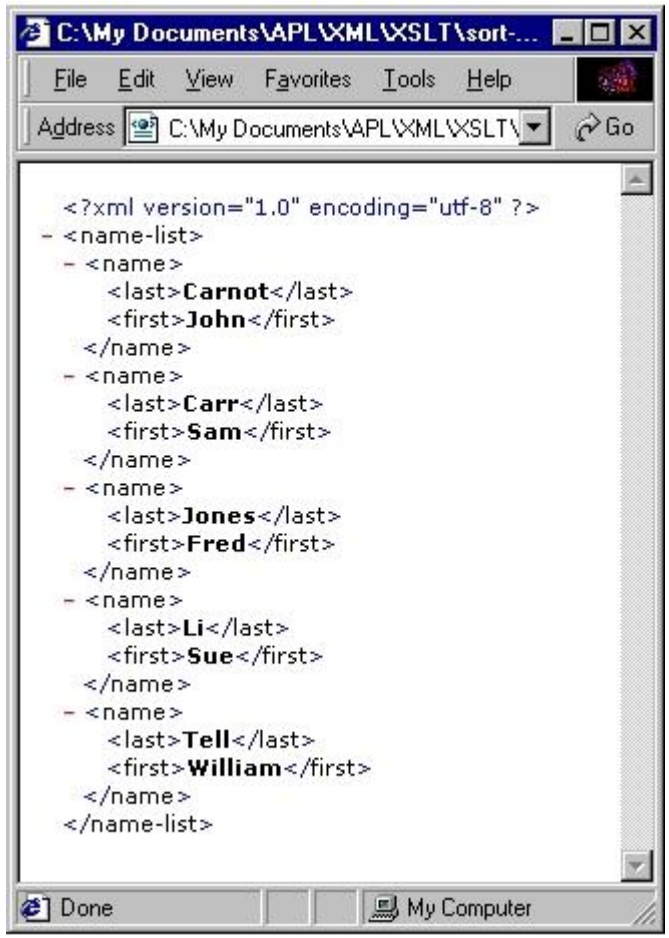
Sort Example

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes"/>

   <xsl:template match="/">
     <name-list>
     <xsl:for-each select="//name">
        <xsl:sort select="last"/>
        <xsl:copy-of select="."/>
     </xsl:for-each>
     </name-list>
   </xsl:template>

</xsl:stylesheet>
```

Sort Output

```
<?xml version="1.0" encoding="utf-8"?>
<name-list>
<name>
        <last>Carnot</last>
        <first>John</first>
    </name>
<name>
        <last>Carr</last>
        <first>Sam</first>
    </name>
<name>
        <last>Jones</last>
        <first>Fred</first>
    </name>
<name>
        <last>Li</last>
        <first>Sue</first>
    </name>
<name>
        <last>Tell</last>
        <first>William</first>
    </name>
</name-list>
```

Sort Display



Modes

- Modes allow one to define different processing for the same tags.
- Both <xsl:template> and <xsl:apply-templates> allow the specification of mode.
  - <xsl:apply-templates select="XPath Expression" mode="*mode*"> is like a call to all templates that are defined with a specific *mode*.
  - <xsl:template select="XPath Expression" mode="*mode*"> defines the mode in which this template is to be called.
- Example modifies customer yet one more time.
  - Prints sorted directory with hyperlinks
  - Prints unsorted list of customers

Customer.xml

```
<customers>
  <good>
    <name>
      <last>Jones</last>
      <first>Fred</first>
      <street>10 Pine St.</street>
      <city>Boise</city>
      <state>ID</state>
      <areacode>208</areacode>
      <exchange>349</exchange>
      <number>2339</number>
    </name>
    <name>
      <last>Li</last>
      <first>Sue</first>
      <street>210 Main St.</street>
      <city>Boston</city>
      <state>MA</state>
      <areacode>617</areacode>
      <exchange>317</exchange>
      <number>5690</number>
    </name>
    <name>
      <last>Carnot</last>
      <first>John</first>
      <street>133 Central Park West</street>
      <city>New York</city>
      <state>NY</state>
      <areacode>212</areacode>
      <exchange>545</exchange>
      <number>9337</number>
    </name>
  </good>
```

Customer.xml (cont.)

```
<bad>
   <name>
      <last>Tell</last>
      <first>William</first>
      <street>33 Ridge Rd. Apt. 3</street>
      <city>Houston</city>
      <state>TX</state>
      <areacode>713</areacode>
      <exchange>864</exchange>
      <number>2006</number>
   </name>
      <name>
         <last>Carr</last>
         <first>Sam</first>
         <street>15 LaCrosse Blvd.</street>
         <city>Chicago</city>
         <state>IL</state>
         <areacode>312</areacode>
         <exchange>395</exchange>
         <number>1087</number>
      </name>
   </bad>
</customers>
```

Directory Example for Modes

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

<xsl:template match="/">
   <HTML>
      <TITLE>Customer Page</TITLE>
      <BODY>
         <H2>Customer Directory:</H2>
         <OL><xsl:apply-templates select="/" mode="directory"/></OL>
         <br/>
         <H2>Customer List:</H2>
         <UL><xsl:apply-templates select="/" mode="list"/></UL>
      </BODY>
   </HTML>
</xsl:template>
```

Modes.xsl (cont.)

```
<xsl:template match="/" mode="directory">
   <xsl:for-each select="//name">
      <xsl:sort select="last"/>
      <LI><A href="{concat('#section-', last)}">
         <xsl:value-of select="last"/></A></LI>
   </xsl:for-each>
</xsl:template>

<xsl:template match="//name" mode="list">
   <LI><A name="{concat('section-', last)}">
      <xsl:value-of select="first"/> <xsl:text> </xsl:text><xsl:value-of select="last"/><br/>
      <xsl:value-of select="street"/><br/>
      <xsl:value-of select="city"/>, <xsl:value-of select="state"/><br/>
      (<xsl:value-of select="areacode"/>) <xsl:text> </xsl:text><xsl:value-of select="exchange"/>-
      <xsl:value-of select="number"/><br/><br/>
      </A></LI>
</xsl:template>
</xsl:stylesheet>
```

Modes Output

```
<HTML>
<TITLE>Customer Page</TITLE>
<BODY>
<H2>Customer Directory:</H2>
<OL>
<LI>
<A href="#section-Carnot">Carnot</A>
</LI>
<LI>
<A href="#section-Carr">Carr</A>
</LI>
<LI>
<A href="#section-Jones">Jones</A>
</LI>
<LI>
<A href="#section-Li">Li</A>
</LI>
```

Modes Output (cont.)

```
<LI>
<A href="#section-Tell">Tell</A>
</LI>
</OL>
<br>
<H2>Customer List:</H2>
<UL>

      <LI>
<A name="section-Jones">Fred Jones<br>10 Pine St.<br>Boise, ID<br>
      (208)  349-
      2339<br>
<br>
</A>
</LI>
```

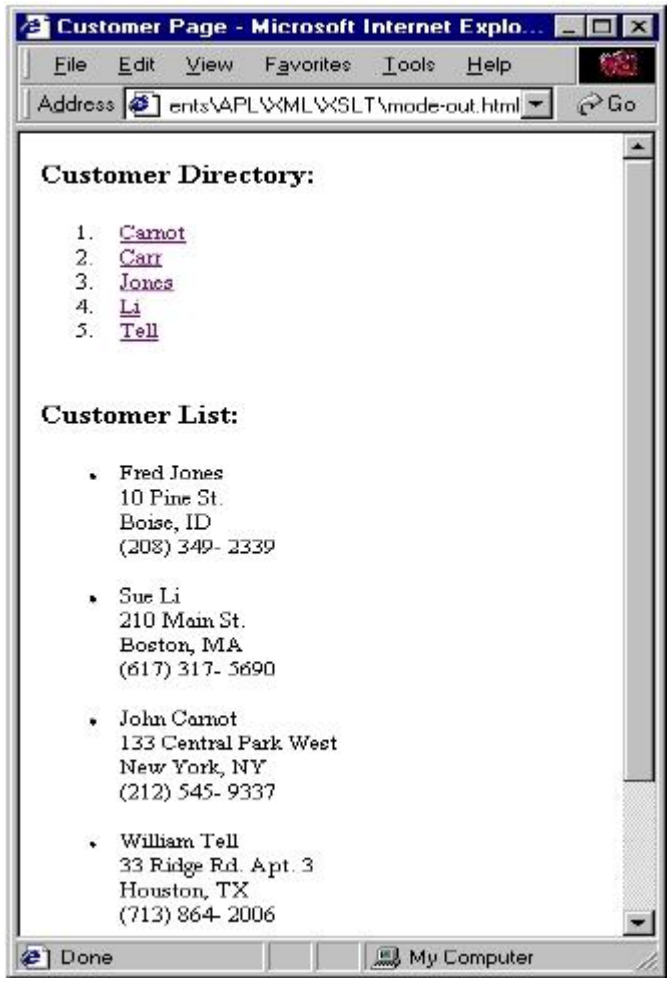
Modes Output (cont.)

```
<LI>
<A name="section-Li">Sue Li<br>210 Main St.<br>Boston, MA<br>
      (617)  317-
      5690<br>
<br>
</A>
</LI>
      <LI>
<A name="section-Carnot">John Carnot<br>133 Central Park West<br>New York, NY<br>
      (212)  545-
      9337<br>
<br>
</A>
</LI>
```

Modes Output (cont.)

```
<LI>
<A name="section-Tell">William Tell<br>33 Ridge Rd. Apt. 3<br>Houston, TX<br>
      (713)  864-
      2006<br>
<br>
</A>
</LI>
      <LI>
<A name="section-Carr">Sam Carr<br>15 LaCrosse Blvd.<br>Chicago, IL<br>
      (312)  395-
      1087<br>
<br>
</A>
</LI>
</UL>
</BODY>
</HTML>
```

Modes Display



Variables

- Can be used as a constant
- A way to name some value to be used multiple times.
- Examples:
  - <xsl:variable name="mgr">Fred</xsl:variable>
  - <xsl:variable name="e">2.71828</xsl:variable>
  - <xsl:variable name="space"><xsl:text> </xsl:text></xsl:variable>
- Uses
  - <xsl:value-of select="$space"/>
  - <xsl:element name="{$mgr}">
- Variables may refer to other variables
- Variable references may not refer back to themselves

Named Templates

- Allows specific templates to be called explicitly
- Example:

```
<xsl:template match="/”>
   <xsl:for-each select="//last">
      <xsl:call-template name=“bold”/>
   </xsl:for-each>
   <xsl:for-each select="//first">
        <xsl:call-template name=“bold”/>
   </xsl:for-each>
</xsl:template>

<xsl:template name=“bold”>
   <B><xsl:value-of select=“.”/></B>
</xsl:template>
```

Parameters

- Templates may not be able to be generic enough
  - We assume that we know the tags names in each case
  - We assume that we are interested in one context at a time
- There are cases in which these assumptions do not hold
- We would also like a general way to print out information from multiply formatted sections of a document
- Example

```
<xsl:template name="name">
    First name: <xsl:value-of select="first"/>
    Last name: <xsl:value-of select="last"/>
</xsl:template>
```

Example

- XML file:

```
<name>
        <first>Jim</first>
        <last>Harris</last>
</name>
<name>
        <first>Sue</first>
        <last>Cheng</last>
</name>
```

- XSL code:

```
<xsl:template name="name-template">
        First name: <xsl:value-of select="first"/>
        Last name: <xsl:value-of select="last"/>
</xsl:template>
```

Parameterized Templates

- What if name elements are not always ‘first’ and ‘last’?
- Use parameterized templates:

```
<xsl:template name="name-template">
    <xsl:param name="first"><xsl:value-of select="first"/>
    </xsl:param>
    <xsl:param name="last"><xsl:value-of select="last"/>
    </xsl:param>

        First name: <xsl:value-of select="$first"/>
        Last name: <xsl:value-of select="$last"/>
</xsl:template>
```

Parameterized Templates (2)

- Values are coming from parameters
  - What did we gain?
  - Values still come from ‘first’ and ‘last’ element values
- However,
  - ‘first’ and ‘last’ element values are only default values
  - We can pass in whatever we wish
  - Template reassigns the parameters with the names ‘$first’ and ‘$last’

Example Use of Parameterized Templates

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

    <xsl:template match="/">
        <xsl:for-each select="//name">
            <xsl:call-template name="name-template"/>
        </xsl:for-each>
    </xsl:template>
    <xsl:template name="name-template">
        <xsl:param name="first"><xsl:value-of select="first"/></xsl:param>
        <xsl:param name="last"><xsl:value-of select="last"/></xsl:param>
            First name: <xsl:value-of select="$first"/>
            Last name: <xsl:value-of select="$last"/>
    </xsl:template>
</xsl:stylesheet>
```

What if the Input is Different?

- XML file:

```
<name>
    <given>Jim</given>
    <surname>Harris</surname>
</name>
<name>
    <given>Sue</given>
    <surname>Cheng</surname>
</name>
```

New XSLT Code

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

    <xsl:template match="/">
        <xsl:for-each select="//name">
            <xsl:call-template name="name-template">
                <xsl:with-param name="first">
                    <xsl:value-of select="given"/>
                </xsl:with-param>
                <xsl:with-param name="last">
                    <xsl:value-of select="surname"/>
                </xsl:with-param>
        </xsl:for-each>
    </xsl:template>
```

New XSLT Code (2)

```
    <xsl:template name="name-template">
        <xsl:param name="first"><xsl:value-of select="first"/></xsl:param>
        <xsl:param name="last"><xsl:value-of select="last"/></xsl:param>
            First name: <xsl:value-of select="$first"/>
            Last name: <xsl:value-of select="$last"/>
    </xsl:template>
</xsl:stylesheet>
```

- The new code produces the correct output even though the element names are different.

Performing “Joins” With XSLT

- Joining two disparate values using XSLT
  - Can be done with variables
  - Can be done with parameterized templates
- Obstacle is dealing with two contexts in the same operation
- Sample student.xml:

```
<?xml version="1.0"?>
<school>
    <students>
        <student>
            <name>Fred</name>
            <course>1</course>
            <course>2</course>
        </student>
```

Student.xml (2)

```
        <student>
            <name>Joe</name>
            <course>2</course>
            <course>3</course>
        </student>
    </students>
    <classes>
        <class number="1">Algebra</class>
        <class number="2">Biology</class>
        <class number="3">Calculus</class>
    </classes>
</school>
```

Joining Student to Class with Variables

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
     <HTML>
        <TITLE>Course Listing</TITLE>
        <BODY>
             <H1>Students and their courses:</H1>
             <OL>
             <xsl:apply-templates/>          <!-- apply the templates for the join -->
             </OL>
        </BODY>
     </HTML>
   </xsl:template>
```

Variable Join Code (2)

```
<xsl:template match="//course">
    <LI>
      <xsl:value-of select="../name"/> --      <!-- print out the name -->
      <xsl:variable name="course-num">    <!-- set the course-num variable -->
          <xsl:value-of select="."/>
      </xsl:variable>

      <xsl:for-each select="//class">            <!-- cycle through the classes -->
      <xsl:if test="number($course-num) = number(./@number)">
          <xsl:value-of select="."/>            <!-- print class name if match occurs -->
      </xsl:if>
    </xsl:for-each>
    </LI>
  </xsl:template>
```

Variable Join Code (3)

```
    <!-- ensure that the names and classes don’t print as a result
         of the default templates -->

    <xsl:template match="//name"/>
    <xsl:template match="//class"/>
</xsl:stylesheet>
```

Joining Student to Class with Parameters

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
     <HTML>
        <TITLE>Course Listing</TITLE>
        <BODY>
             <H1>Students and their courses:</H1>
             <OL>
             <xsl:apply-templates/>          <!-- apply the templates for the join -->
             </OL>
        </BODY>
     </HTML>
   </xsl:template>
```

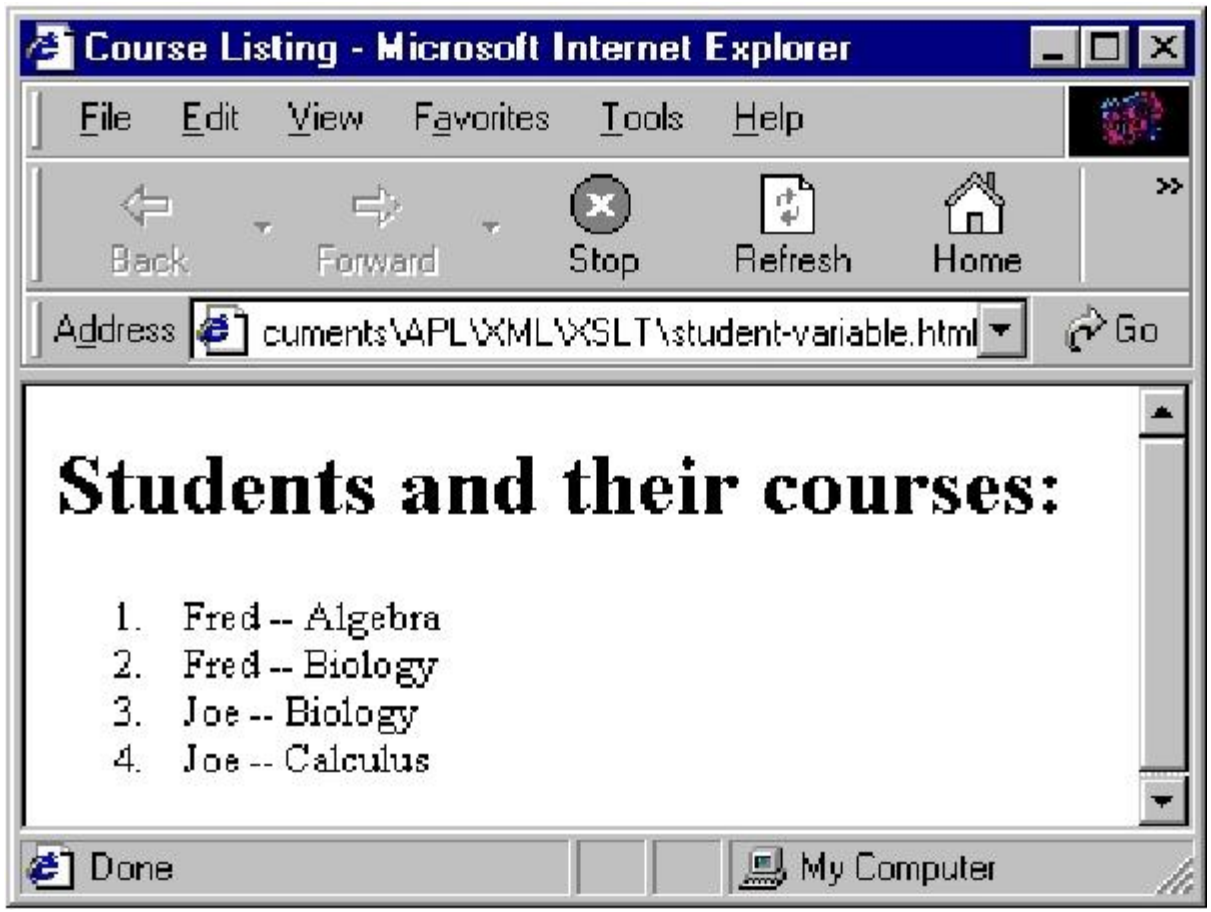
Parameterized Template Join Code (2)

```
<xsl:template match="//course">
  <LI>
     <xsl:value-of select="../name"/> --          <!-- print out the name -->
     <xsl:call-template name="course-match"/>     <!-- call “course-match” template -->
  </LI>
</xsl:template>

<xsl:template name="course-match">
  <xsl:param name="course-num"><xsl:value-of select="."/></xsl:param>
  <xsl:for-each select="//class">                 <!-- cycle through the classes -->
     <xsl:if test="number($course-num) = number(./@number)">
        <xsl:value-of select="."/>                <!-- print class name if matches -->
     </xsl:if>
  </xsl:for-each>
</xsl:template>
```

Parameterized Template Join Code (3)

```
    <!-- ensure that the names and classes don’t print as a result
         of the default templates -->

    <xsl:template match="//name"/>
    <xsl:template match="//class"/>
</xsl:stylesheet>
```

Output


Course Listing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address cuments\APL\XML\XSLT\student-variable.html
Go
Students and their courses:
1. Fred -- Algebra
2. Fred -- Biology
3. Joe -- Biology
4. Joe -- Calculus
Done
My Computer

Using XSLT with CSS

- You cannot use CSS to pull information from XML documents.
  - If display is dependent on information within XML document, CSS by itself cannot be used
- CSS cannot reorder values.
- CSS is not interactive or dynamic.
  - You cannot write XSLT-like programs
- CSS is useful for display
- Both XSLT and CSS can be used together.
- For example, print the good customers in blue and the bad customers in red.

Directory Example with CSS

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <HTML>
      <TITLE>Customer Page</TITLE>
      <STYLE>
         LI.good {color:blue;}
         LI.bad  {color:red;}
      </STYLE>
      <BODY>
         <H2>Customer Directory:</H2>
         <OL><xsl:apply-templates select="/" mode="directory"/></OL><br/>
         <H2>Customer List:</H2>
         <UL><xsl:apply-templates select="/" mode="list"/></UL>
      </BODY>
   </HTML>
</xsl:template>
```

Directory Example with CSS (cont.)

```
<xsl:template match="/" mode="directory">
   <xsl:for-each select="//name">
      <xsl:sort select="last"/>
      <LI><A href="{concat('#section-', last)}">
         <xsl:value-of select="last"/></A></LI>
   </xsl:for-each>
</xsl:template>
<xsl:template match="//name" mode="list">
   <LI class="{name(..)}"><A name="{concat('section-', last)}">
      <xsl:value-of select="first"/><xsl:text> </xsl:text><xsl:value-of select="last"/><br/>
      <xsl:value-of select="street"/><br/>
      <xsl:value-of select="city"/>, <xsl:value-of select="state"/><br/>
      (<xsl:value-of select="areacode"/>) <xsl:text> </xsl:text><xsl:value-of select="exchange"/>-
      <xsl:value-of select="number"/><br/><br/>
      </A></LI>
</xsl:template>
</xsl:stylesheet>
```

Directory Example with CSS - Output

```
<HTML>
<TITLE>Customer Page</TITLE>
<STYLE>
        LI.good {color:blue;}
        LI.bad  {color:red;}
      </STYLE>
<BODY>
<H2>Customer Directory:</H2>
<OL>
<LI>
<A href="#section-Carnot">Carnot</A>
</LI>
<LI>
<A href="#section-Carr">Carr</A>
</LI>
<LI>
<A href="#section-Jones">Jones</A>
</LI>
```

Directory Example with CSS - Output (cont.)

```
<LI>
<A href="#section-Li">Li</A>
</LI>
<LI>
<A href="#section-Tell">Tell</A>
</LI>
</OL>
<br>
<H2>Customer List:</H2>
<UL>

    <LI class="good">
<A name="section-Jones">Fred Jones<br>10 Pine St.<br>Boise, ID<br>
    (208)  349-
    2339<br>
<br>
</A>
```

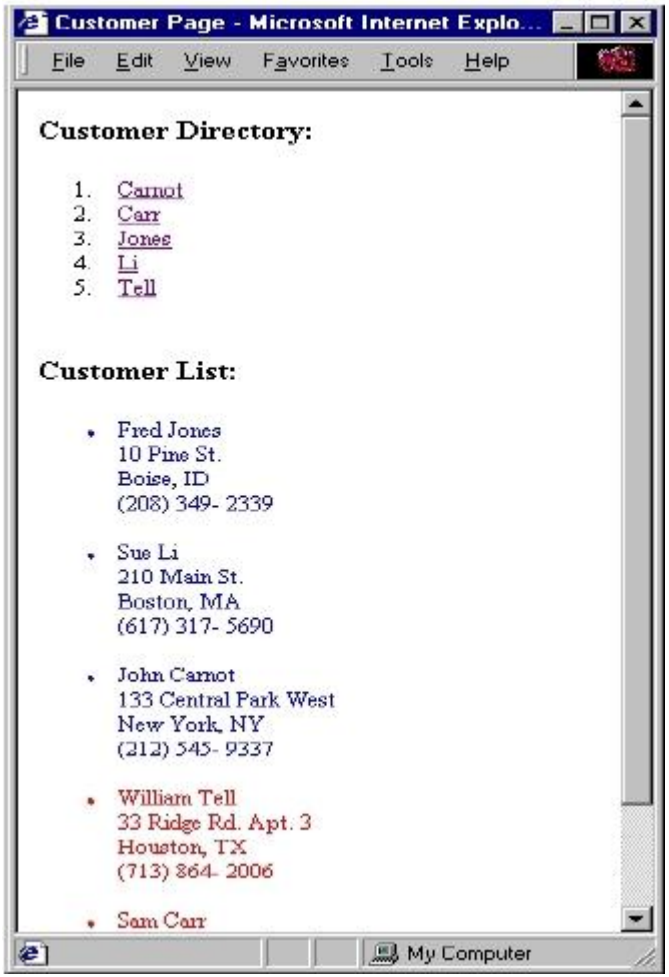
Directory Example with CSS - Output (cont.)

```
</LI>
      <LI class="good">
<A name="section-Li">Sue Li<br>210 Main St.<br>Boston, MA<br>
      (617)  317-
      5690<br>
<br>
</A>
</LI>
      <LI class="good">
<A name="section-Carnot">John Carnot<br>133 Central Park West<br>New York, NY<br>
      (212)  545-
      9337<br>
<br>
</A>
</LI>
```

Directory Example with CSS - Output (cont.)

```
        <LI class="bad">
<A name="section-Tell">William Tell<br>33 Ridge Rd. Apt. 3<br>Houston, TX<br>
        (713)  864-
        2006<br>
<br>
</A>
</LI>
        <LI class="bad">
<A name="section-Carr">Sam Carr<br>15 LaCrosse Blvd.<br>Chicago, IL<br>
        (312)  395-
        1087<br>
<br>
</A>
</LI>

</UL>
</BODY>
</HTML>
```

Browser Output



Τέλος Ενότητας