Freie Universität Berlin

Musterlösung des Übungsblattes 4

Beispieltransformation

```
<document xmlns="http://www.doc.org/ns" …>
        <title>Beginning XML</title>
        <author>Hunter et al.</author>
        <chapter name="Chapter 1">
                <section>Chap. 1_1</section>
                <section>Chap. 1_2</section>
                <section>Chap. 1_3</section>
        </chapter>
        <chapter name="Chapter 2">
                <section>Chap. 2_1</section>
                <section>Chap. 2_2</section>
        </chapter>
        <chapter name="Chapter 3">
                <section>Chap. 3_1</section>
                <section>Chap. 3_2</section>
                <section>Chap. 3_3</section>
                <section>Chap. 3_4</section>
        </chapter>
</document>
```

Beginning XML

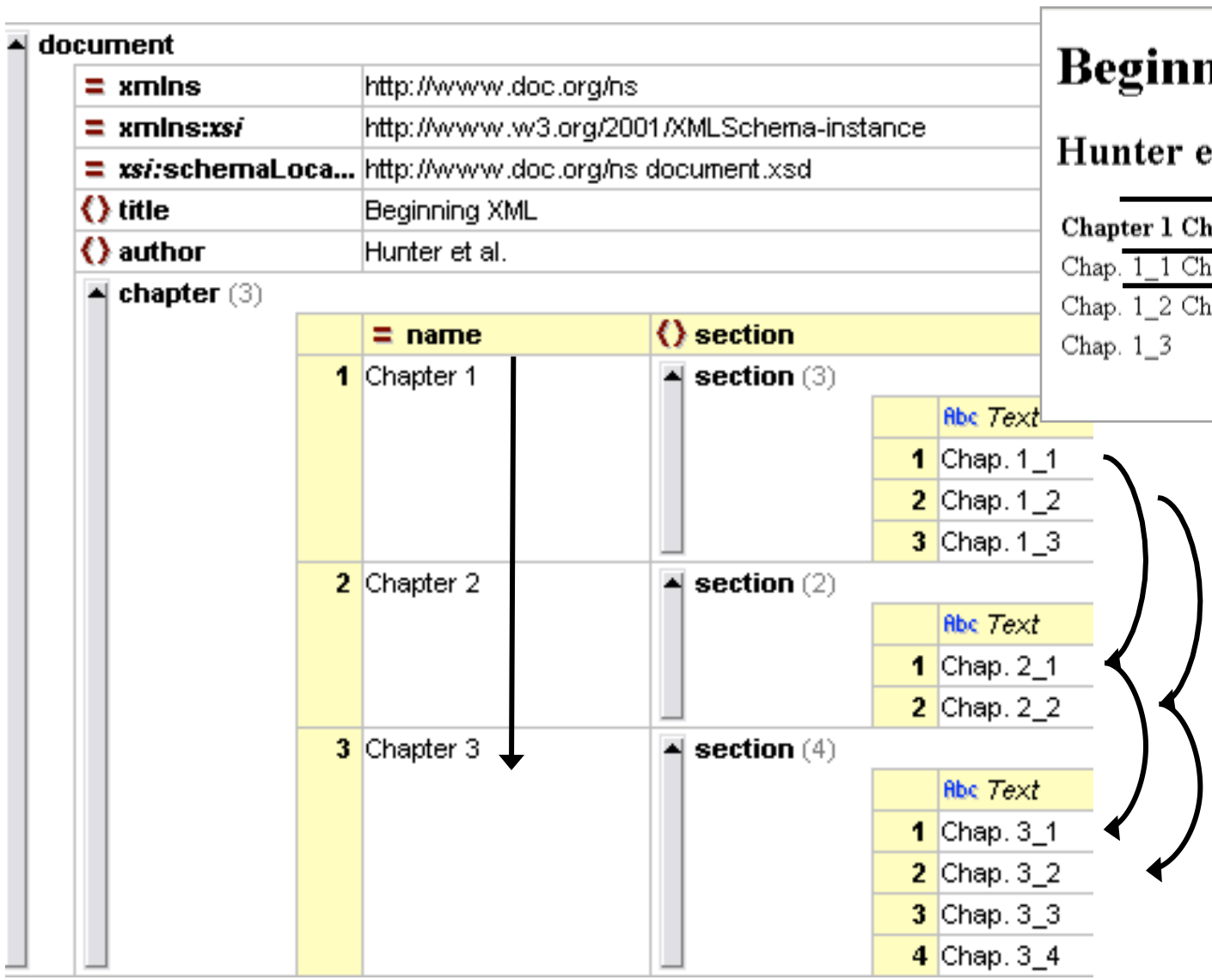
Hunter et. al

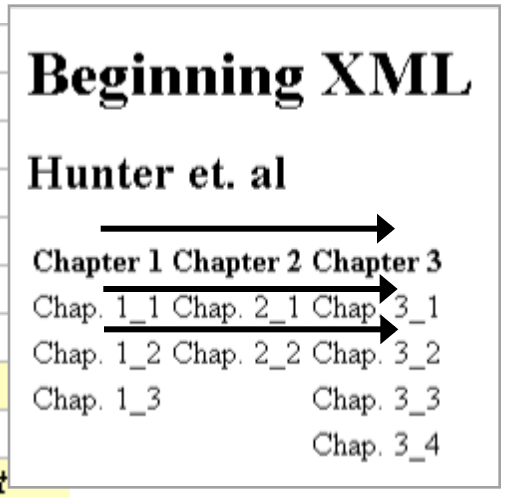
Chapter 1	Chapter 2	Chapter 3
Chap. 1_1	Chap. 2_1	Chap. 3_1
Chap. 1_2	Chap. 2_2	Chap. 3_2
Chap. 1_3		Chap. 3_3
		Chap. 3_4

Ergebnisdokument

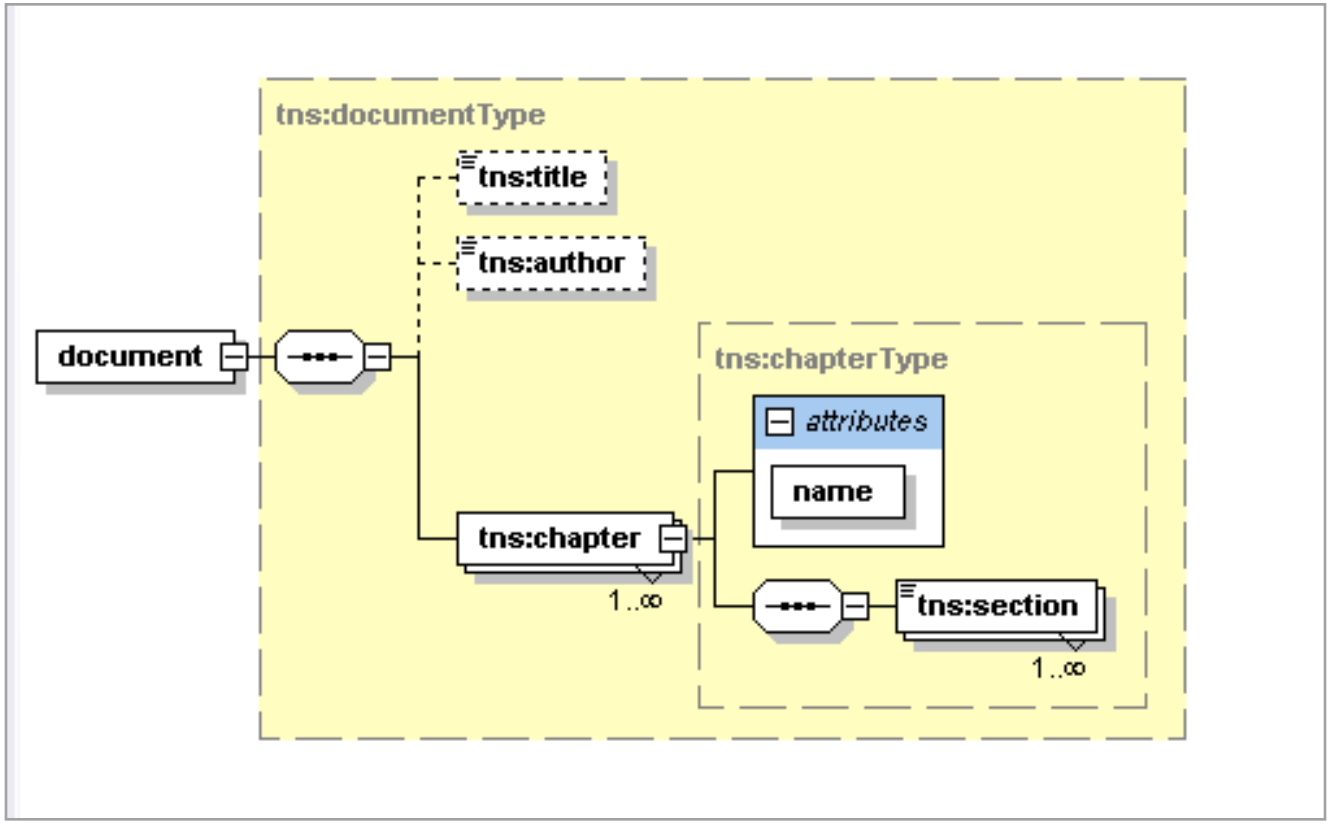
```
 1  <html>
 2    <head>
 3      <title />
 4    </head>
 5    <body>
 6      <h1>Beginning XML</h1>
 7      <h2>Hunter et. al</h2>
 8      <TABLE>
 9        <TR>
10          <TH>Chapter 1</TH>
11          <TH>Chapter 2</TH>
12          <TH>Chapter 3</TH>
13        </TR>
14        <TR>
15          <TD>Chap. 1_1 </TD>
16          <TD>Chap. 2_1 </TD>
17          <TD>Chap. 3_1 </TD>
18        </TR>
19        <TR>
20          <TD>Chap. 1_2 </TD>
21          <TD>Chap. 2_2 </TD>
22          <TD>Chap. 3_2 </TD>
23        </TR>
24        <TR>
25          <TD>Chap. 1_3 </TD>
26          <TD> </TD>
27          <TD>Chap. 3_3 </TD>
28        </TR>
29        <TR>
30          <TD> </TD>
31          <TD> </TD>
32          <TD>Chap. 3_4 </TD>
33        </TR>
34      </TABLE>
35    </body>
36  </html>
```

Nötige strukturelle Transformation





Klasse der Ursprungsdokumente



Grundstruktur der Lösung

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet version="1.0"
3   xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ns="
    http://www.doc.org/ns">
4       <xsl:output method="html"/>
5       <xsl:template match="ns:document">
6           <html>
23      </xsl:template>
24      <xsl:template name="rowCounter">
40      <xsl:template match="ns:title">
45      <xsl:template match="ns:author">
50      <xsl:template match="ns:chapter">
55  </xsl:stylesheet>
```

- für jedes Element außer ns:section ein Template
- match="ns:document": alle document–Elemente, die Namensraum http://www.doc.org/ns zugeordnet sind

Erzeugung der HTML-Grundstruktur

```
5   <xsl:template match="ns:document">
6       <html>
7           <head>
8               <title/>
9           </head>
10          <body>
22      </html>
23  </xsl:template>
```

Transformation von title und author

```
 5   <xsl:template match="ns:document">
 6     <html>
 7       <head>
 8         <title/>
 9       </head>
10       <body>
11         <xsl:apply-templates select="ns:title"/>
12         <xsl:apply-templates select="ns:author"/>
13         <TABLE>
21       </body>
22     </html>
23   </xsl:template>
24   <xsl:template name="rowCounter">
40   <xsl:template match="ns:title">
45   <xsl:template match="ns:author">
50   <xsl:template match="ns:chapter">
```

- Was passiert, wenn title oder author nicht vorhanden ist?

ns:title ➔ <h1>, ns:author ➔ <h2>

```
40  <xsl:template match="ns:title">
41    <h1>
42      <xsl:value-of select="."/>
43    </h1>
44  </xsl:template>
45  <xsl:template match="ns:author">
46    <h2>
47      <xsl:value-of select="."/>
48    </h2>
49  </xsl:template>
```

- Was passiert, wenn title oder author Kind-Elemente haben?

Select . or text() ?

```
<xsl:template match="p">
        <DIV>
          <xsl:copy-of select="."/>
        </DIV>
        <DIV>
          <xsl:copy/>
        </DIV>
        <DIV>
          <xsl:value-of select="."/>
        </DIV>
</xsl:template>
```

```
<source>
    <p id="a12">Compare
        <B>these constructs</B>.
    </p>
</source>
```

```
<DIV>
  <p id="a12">Compare
    <B>these constructs</B>.
  </p>
</DIV>
<DIV>
  <p/>
</DIV>
```

```
<DIV>
  Compare these constructs.
</DIV>
```

oder mit "text()" statt "."

```
<DIV>
  Compare
</DIV>
```

Spaltenüberschriften erzeugen

Freie Universität Berlin

Beginning XML

Hunter et. al

Chapter 1	Chapter 2	Chapter 3
Chap. 1_1	Chap. 2_1	Chap. 3_1
Chap. 1_2	Chap. 2_2	Chap. 3_2
Chap. 1_3		Chap. 3_3
		Chap. 3_4

chapter (3)

	= name	{} section
1	Chapter 1	**section** (3)
2	Chapter 2	**section** (2)
3	Chapter 3	**section** (4)

	Abc *Text*
1	Chap. 1_1
2	Chap. 1_2
3	Chap. 1_3

	Abc *Text*
1	Chap. 2_1
2	Chap. 2_2

	Abc *Text*
1	Chap. 3_1
2	Chap. 3_2
3	Chap. 3_3
4	Chap. 3_4

Grundstruktur der Tabelle erzeugen

```
 5   <xsl:template match="ns:document">
 6     <html>
 7       <head>
10       <body>
11         <xsl:apply-templates select="ns:title"/>
12         <xsl:apply-templates select="ns:author"/>
13         <TABLE>
14           <TR>
15             <xsl:apply-templates select="ns:chapter"/>
16           </TR>
17           <xsl:call-template name="rowCounter">
20         </TABLE>
21       </body>
22     </html>
23   </xsl:template>
```

- Beachte: select="ns:chapter" meint alle Kind-Elemente ns:chapter

Spaltenüberschriften <TH> erzeugen

```
50    <xsl:template match="ns:chapter">
51      <TH>
52        <xsl:value-of select="@name"/>
53      </TH>
54    </xsl:template>
```

Zeilen <TR> erzeugen

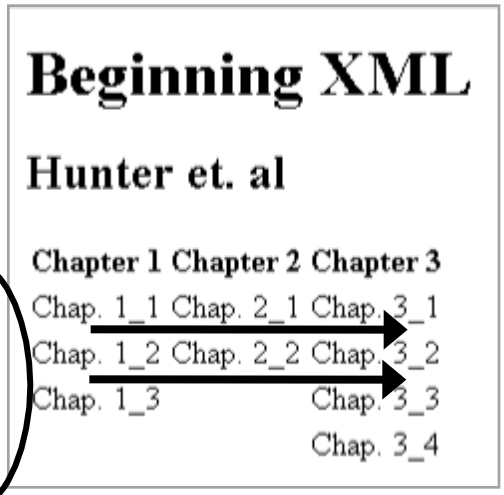
chapter (3)

	= name	{} section
1	Chapter 1	section (3)
2	Chapter 2	section (2)
3	Chapter 3	section (4)

	Abc Text
1	Chap. 1_1
2	Chap. 1_2
3	Chap. 1_3

	Abc Text
1	Chap. 2_1
2	Chap. 2_2

	Abc Text
1	Chap. 3_1
2	Chap. 3_2
3	Chap. 3_3
4	Chap. 3_4



Für N von 1 bis zur maximalen Anzahl von sections:

Erzeuge <TR> mit <TD>s, die Nte section von jedem chapter enthalten!

Erzeuge Zeile N!

```
24  <xsl:template name="rowCounter">
25    <xsl:param name="N"/>
26    <xsl:if test="ns:chapter/ns:section[ $N ]">
27      <TR>
28        <xsl:for-each select="ns:chapter">
29          <TD>
30            <xsl:apply-templates select="ns:section[ $N ]"/>
31            <xsl:text> </xsl:text>
32          </TD>
33        </xsl:for-each>
34      </TR>
35      <xsl:call-template name="rowCounter">
38    </xsl:if>
39  </xsl:template>
```

Erzeuge Zeile N!

```
24  <xsl:template name="rowCounter">
25    <xsl:param name="N"/>
26    <xsl:if test="ns:chapter/ns:section[ $N ]">
27      <TR>
28        <xsl:for-each select="ns:chapter">
29          <TD>
30            <xsl:apply-templates select="ns:section[ $N ]"/>
31            <xsl:text> </xsl:text>
32          </TD>
33        </xsl:for-each>
34      </TR>
35      <xsl:call-template name="rowCounter">
38    </xsl:if>
39  </xsl:template>
```

- Beachte: für ns:section kein Template definiert
- Was passiert hier ohne ein solches Template?

Vordefinierte Templates

1. vordefiniertes Template

- realisiert rekursiven Aufruf des Prozessors, wenn kein Template anwendbar ist:

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

2. vordefiniertes Template

- kopiert PCDATA und Attribut-Werte des aktuellen Knotens in das Ergebnisdokument:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

Vermeide <TD/> !

```
24  <xsl:template name="rowCounter">
25    <xsl:param name="N"/>
26    <xsl:if test="ns:chapter/ns:section[ $N ]">
27      <TR>
28        <xsl:for-each select="ns:chapter">
29          <TD>
30            <xsl:apply-templates select="ns:section[ $N ]"/>
31            <xsl:text> </xsl:text>
32          </TD>
33        </xsl:for-each>
34      </TR>
35      <xsl:call-template name="rowCounter">
38    </xsl:if>
39  </xsl:template>
```

selbstschließende Elemente <TD/> vermeiden,
da HTML <TD></TD> erwartet

Rekursion: Erzeuge Zeile N+1

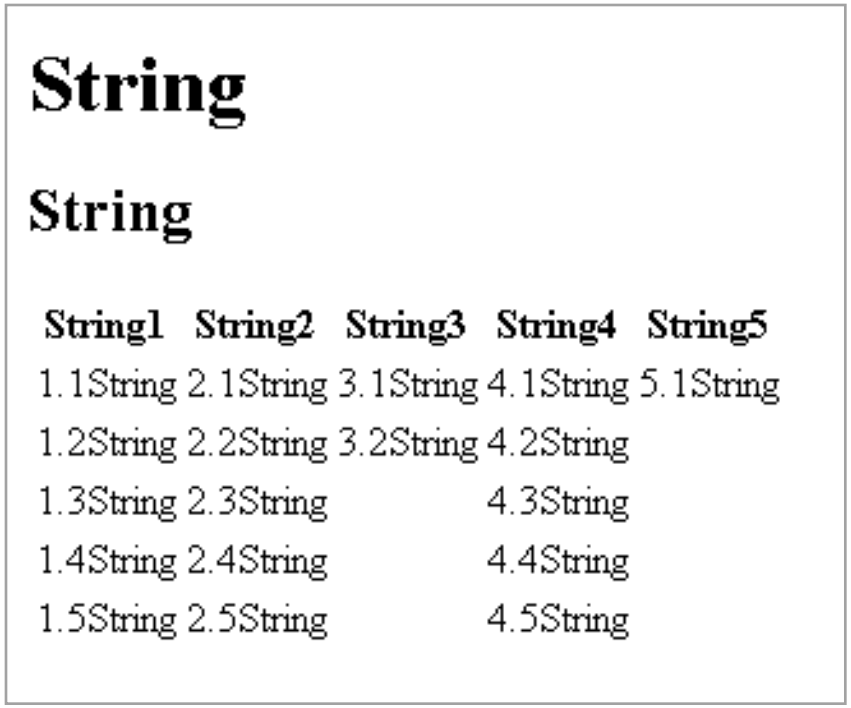
```
24  <xsl:template name="rowCounter">
25    <xsl:param name="N"/>
26    <xsl:if test="ns:chapter/ns:section[ $N ]">
27      <TR>
28        <xsl:for-each select="ns:chapter">
29          <TD>
30            <xsl:apply-templates select="ns:section[ $N ]"/>
31            <xsl:text> </xsl:text>
32          </TD>
33        </xsl:for-each>
34      </TR>
35      <xsl:call-template name="rowCounter">
36        <xsl:with-param name="N" select="$N + 1"/>
37      </xsl:call-template>
38    </xsl:if>
39  </xsl:template>
```

Und der eigentliche Aufruf mit N=1

```
 5  <xsl:template match="ns:document">
 6    <html>
 7      <head>
10      <body>
11        <xsl:apply-templates select="ns:title"/>
12        <xsl:apply-templates select="ns:author"/>
13        <TABLE>
14          <TR>
17          <xsl:call-template name="rowCounter">
18            <xsl:with-param name="N" select="1"/>
19          </xsl:call-template>
20        </TABLE>
21      </body>
22    </html>
23  </xsl:template>
```

Wie testen?

1. XML-Schema: automatisch Instanz generieren
2. generierte Instanz editieren
3. Instanz transformieren



String

String

String1	String2	String3	String4	String5
1.1String	2.1String	3.1String	4.1String	5.1String
1.2String	2.2String	3.2String	4.2String	
1.3String	2.3String		4.3String	
1.4String	2.4String		4.4String	
1.5String	2.5String		4.5String	

Freie Universität Berlin

Musterfragen

Frage 1

Freie Universität Berlin

Which of the following XPath expressions will select all shoe children of the current node that have a width attribute with the value of "EEEE"?

A. shoe[attribute::width="EEEE"]

B. shoe[@width="EEEE"]

C. shoe(@width="EEEE")

D. child::shoe[attribute::width="EEEE"]

E. child::shoe(attribute::width="EEEE")

Frage 4

```
<?xml version="1.0" encoding="UTF-8"?>
<periodicTable>
  <chemicalElement symbol="Ag">
    <atomicNumber>47</atomicNumber>
    <atomicWeight>107.8682</atomicWeight>
  </chemica
  <chemica
    <atomic
    <atomic
  </chemica
</periodicT
```

Which XPath expression would select just the child elements of all chemicalElement elements of the periodicTable element?

A. /periodicTable/chemicalElement*

B. /periodicTable/*

C. /*/*/*

D. //*

Frage 5

Freie Universität Berlin

Which XPath expression selects all species elements that have a mutation element?

A. species[mutation]

B. species(mutation)

C. species/mutation

D. species[@mutation]

Frage 6

Freie Universität Berlin

Which XPath expression selects the first manifold node that has a riemann attribute?

A. manifold[@riemann[1]]

B. manifold[position()=1][@riemann]

C. manifold[1]/[@riemann]

D. manifold[@riemann][1]

E. [1]manifold[@riemann]

Multiple Xpath predicates

Beware of one trap when using the approach employed in the first of the three preceding location paths -- I think of it as a "stacked" predicate. The order in which predicates appear on the stack can affect the final result. Each succeeding predicate is evaluated in terms of the narrowed context provided by the preceding one(s), and not just in terms of the general context in which a single predicate would be evaluated. Here's a sample document to illustrate this point.

```
<tosses>
   <toss result="heads"/>
   <toss result="heads"/>
   <toss result="tails"/>
   <toss result="heads"/>
</tosses>
```

Now consider the following two location paths into this document, each using a stacked predicate:

```
(//toss)[@result="heads"][3]
(//toss)[3][@result="heads"]
```

See the difference? The first path locates (a) all **toss** elements whose **result** attribute equals "heads", and then (b) the third one of those **toss** elements. Therefore, it selects the fourth **toss** element in the document.

The second path selects the third **toss** element, and then the stacked predicate applies a further screen, selecting the third **toss** element only if its **result** attribute has a value of "heads". Because the third **toss** element's **result** attribute is "tails", therefore, this location path returns an empty node-set.

Frage 7

Which XPath expression selects the price of Borshch?

A. //entree[@cuisine='Borshch']/price

B. //entree[dish='Borshch']/price

C. //entree[@dish='Russian']/price

D. //entree[text()='Borshch']/price

E. //entree['Borshch']/price

```
<?xml version="1.0" encod
<menu>
  <entree cuisine="Cajun">
    <dish>Jambalaya</dish
    <price denomination="
  </entree>
  <entree cuisine="Russian">
    <dish>Borshch</dish>
    <price denomination="ruble">6.66</price>
  </entree>
</menu>
```

Noch ein kleiner Tip zu CDATA...

Freie Universität Berlin

Note that attribute value literals are always parsed as *replaceable character data*, regardless of the attribute's declared value. This means that references (`&xxx;`, `&#yyy;`) are recognized and replaced in attribute specifications, *even for* `CDATA` *attributes*.

For example, this HTML fragment:

```
<IMG SRC="eqn1.gif" ALT = "A &lt; B">
```

will be displayed as

```
A < B
```

in a text-mode browser or with image loading turned off (assuming the browser is working properly, of course).

Freie Universität Berlin

XSLT in Depth

Valid XSLT

```
<xsl:stylesheet> </xsl:stylesheet>
```

```
<xsl:stylesheet     xmlns:xsl=„http://
www.w3.org/1999/XSL/
     Transform“

>

</xsl:stylesheet>
```

```
<xsl:stylesheet version=„1.0“

     xmlns:xsl=„http://www.w3.org/1999/XSL/
                    Transform“

>

</xsl:stylesheet>
```

Valid XSLT

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

```
<name>
     <first_name>John</first_name>
     <last_name>Smith</last_name>
</name>
```

Valid XSLT

```
John

Smith
```

How come?

Never forget about the….

- Default templates
  - Are ALWAYS used by the XSLT processor
  - Can‘t be switched off, only OVERRIDDEN

```
<xsl:template match=„*|/“

     <xsl:apply-templates/>

</xsl:template>
```

```
<xsl:template match=„text()|@*“

     <xsl:value-of select=„.“/>

</xsl:template>
```

```
<xsl:template match=„processing-instruction
()|comment()“ />
```

Overriding templates

- If a match exists in the stylesheet, that template will „override“ the default template

```
<xsl:template match=„name“>
     <!-- Do Nothing -->
</xsl:template>
```

overrides

```
<xsl:template match=„*|/“
     <xsl:apply-templates/>
</xsl:template>
```

Overriding templates

- <xsl:template match="/"> … </xsl:template>

- Sobald ihr das obenstehende nutzt, und NICHT rekursiv <xsl:apply-templates /> aufruft innerhalb des xsl:templates, ist das untenstehende Default-Template überschrieben und wird nicht mehr ausgeführt.

```
<xsl:template match=„*|/“
      <xsl:apply-templates/>
</xsl:template>
```

Overriding templates

- Experimentiert bis ihr es verstanden habt!

http://w3schools.com/xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog

Overriding templates (2)

- So what about this? Does the more specific template override the more general?

```
<xsl:template match=„name“>
     <!-- Do Nothing -->
</xsl:template>
```

is overriden by

```
<xsl:template match=„name[2]“
     <xsl:apply-templates/>
</xsl:template>
```

Overriding templates (3)

- And what about this? What happens if the name element in the second position has also a lang attribute with the value ‚en'?

```
<xsl:template match=„name[@lang=‚en']"">
     <!-- Do Nothing -->
</xsl:template>
```

???

```
<xsl:template match=„name[2]"
     <xsl:apply-templates/>
</xsl:template>
```

Overriding templates (3)

Mehrere Templates die auf dasselbe Element matchen aber mit unterschiedlichen gleichpriorisierten Kriterien sind ein Zeichen schlechten Programmierstils!

Stattdessen 1x auf das Element matchen

➔Dann innerhalb des Templates bedingte Ausgaben

<xsl:if test="price > 10">

oder

<Xsl:choose> mit mehreren

<xsl:when>….

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

Template priority

Freie Universität Berlin

- „If more than one template rule with the same import precedence matches a given node....“
  - Import precedence is default template rules, explicitly imported rules and then rules in the actual stylesheet

- „... the one with the highest priority is chosen“
  - Rules with match patterns of a single element or attribute name have priority 0
  - Rules with match patterns of *prefix:** have priority -0.25
  - Rules with match patterns which have only a wildcard test (*,@*,node() ...) have priority -0.5
  - Rules with any other patterns, e.g. with location paths or predicates, have priority 0.5

Template priority (2)

Freie Universität Berlin

- „It is an error if two or more template rules match a node and have the same priority“
  - Most XSLT processors choose the LAST template rule rather than signal an error

- One can explicitly give a template a priority

```
<xsl:template match=„…“ priority=„1“>
```

- How to execute multiple rules on the same node?
  - Use a more general match on a template rule
  - Within the rule, make the specific node selection
  - You can use named templates rather than matching on the selected node

Overriding by import

- You can use

```
<xsl:import href="import.xsl"/>
```

- However, imported templates have less precedence than templates in the main stylesheet
- To „force“ the processor to use BOTH an imported template and a template in the main stylesheet:

```
<xsl:apply-imports />
```

- The problem is that imports must be top level elements and come before all other elements, hence are always overridden
- The alternative is to include templates at the top level but ANYWHERE in the stylesheet, such as right at the end

```
<xsl:include href=„import.xsl“/>
```

XSLT and namespaces

- XSLT processors are namespace-aware
  - Patterns must identify namespaces of elements

```
<name xmlns=„http://www.name.org“>
      <first_name>John</first_name>
      <last_name>Smith</last_name>
</name>
```

```
<xsl:template match=„name“>
      wird dieser Text ausgegeben oder nicht?
</xsl:template>
```

XSLT and namespaces

- XSLT processors are namespace-aware
  - Patterns must identify namespaces of elements

```
<name xmlns=„http://www.name.org“>
      <first_name>John</first_name>
      <last_name>Smith</last_name>
</name>
```

```
<xsl:template match=„name“>
      wird dieser Text ausgegeben oder nicht? NEIN!
</xsl:template>
```

XSLT and namespaces

```
<name xmlns=„http://www.name.org“>
     <first_name>John</first_name>
     <last_name>Smith</last_name>
</name>
```

```
<xsl:stylesheet xmlns=„http://
www.name.org“>
     <xsl:template match=„name“>
          sorry, so klappt es NICHT!
     </xsl:template>
</xsl:stylesheet>
```

XSLT and namespaces

```
<aa:name xmlns:aa=„http://www.name.org“>
     <aa:first_name>John</aa:first_name>
     <aa:last_name>Smith</aa:last_name>
</aa:name>
```

```
<xsl:stylesheet                    xmlns:zz=„http://
www.name.org“>
     <xsl:template match=„zz:name“>
          ich werde ausgegeben!
     </xsl:template>
</xsl:stylesheet>
```