Übung 2

xsl:for-each, xsl:if, xsl:choose, xsl:when

<xsl:for-each>

In Übung 1 wurden mit `<xsl:apply-templates/>` alle Knoten angesprochen und ohne weitere Anweisungen ausgegeben. Jetzt wollen wir jedoch die Literaturliste auch als Liste ausgeben. Mit `<xsl:for-each>` kann man innerhalb eines Templates eine Schleife über ausgewählte Knoten legen. Damit werden mehrere aufeinanderfolgende Knoten hintereinander abgearbeitet. So kann man innerhalb einer Template Anweisung mehrere Elemente auf einmal verarbeiten.

A. Lege einen Ordner “Übung2” an und speichere darin ein neu angelegtes Stylesheet “Übung2.xsl”. Das Quelldokument ist wieder “Übung_booklist.xml”.
B. Wir verwenden eine HTML Liste. Die Liste wird im html-Element <ul> (unordered list) mit den Kindelementen <li> (list) strukturiert. Zunächst erstellen wir ein Template fuer das Rootelement des Quelldokumentes und legen darin ein HTML Liste an. Schreib das folgende Template

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      exclude-result-prefixes="xs"
5      version="2.0">
6
7      <xsl:template match="bookList">
8          <ul></ul>
9      </xsl:template>
10
11
12
13 </xsl:stylesheet>
14
```

C. Anstelle alle Knoten mit dem leeren Element `<xsl:apply-templates/>` auszulesen, legen wir jetzt eine Schleife an. Die Schleife soll aus jedem `<bibl>` Element den Autor in ein Listenelement auslesen.
Innerhalb von `<ul>` schreiben wir dafür eine `For-Each`-Schleife und wählen das Element das bei der Abfrage immer wieder aufgerufen werden soll `<xsl:for-each select=”bibl”>`. Unser For-each-Befehl sagt dem Prozessor, dass er über alle `<bibl>` Elemente, die Kindelemente von `<bookList>` sind, iterieren soll.

```
<xsl:template match="bookList">
      <ul>
            <xsl:for-each select=”bibl”>
            </xsl:for-each>
```

```
      </ul>
</xsl:template>
```

D. Die obere `For-Each`-Schleife hat jedoch keinen Effekt (und es wird nichts ausgegeben), da im Schleifenkörper keine weiteren Anweisungen definiert sind.
E. Als nächstes definieren wir weitere Befehle für die Ausgabe von Kindelementen von `<bibl>`. Im Beispiel unten sollen alle Autoren, die gefunden werden, in einem Listenelement `<li>` ausgegeben werden. Mit dem Element `<xsl:value-of select=””>` wählt man den Knoten, dessen Inhalt ausgelesen werden soll.

```
<xsl:template match="bookList">
   <ul>
      <xsl:for-each select=”bibl”>
         <li>
            <xsl:value-of select=”author”/>
          </li>
       </xsl:for-each>
    </ul>
 </xsl:template>
```

Das Stylesheet sollte wie folgt aussehen:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      exclude-result-prefixes="xs"
5      version="2.0">
6
7      <xsl:template match="bookList">
8          <ul>
9              <xsl:for-each select="bibl">
10                 <li>
11                     <xsl:value-of select="author"/>
12                 </li>
13             </xsl:for-each>
14         </ul>
15     </xsl:template>
16
17 </xsl:stylesheet>
18
```

F. Führe eine Transformation wie in Übung 1 aus und schau dir das Ergebnis im Browser an.
G. Füge eine Überschrift `<h1>Literaturliste</h1>` vor der Literaturliste und den Titel zu jedem Eintrag hinzu.
H. Hinweis: Mit dem Element `<xsl:text>` können Text, Interpunktion und Leerzeichen hinzugefügt werden (siehe z.B.: https://www.w3schools.com/xml/ref_xsl_el_text.asp).

```xml
<?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:template match="bookList">
        <h1>Literaturliste</h1>
        <ul>
            <xsl:for-each select="bibl">
                <li>
                    <xsl:value-of select="author"/>
                    <xsl:text>: </xsl:text>
                    <xsl:value-of select="title"/>
                </li>
            </xsl:for-each>
        </ul>
    </xsl:template>

</xsl:stylesheet>
```

I. Nachdem die Transformation nochmals ausgeführt wurde, sollte das Ergebnis wie folgt aussehen:

Literaturliste

- Doug Tidwell: XSLT
- John Simpson: XPath and XPointer Locating Content in XML Documents
- James Joyce: Ulysses
- Michael Kay: XSLT 2.0 and XPath 2.0 Programmer's Reference, 4th Edition
- Jim Melton Stephen Buxton: Querying XML
- Johann Wolfgang von Goethe: Faust A Dramatic Poem

xsl:choose

Wie man am Ergebnis sehen kann, stehen die Titel und die Untertitel der Bücher hintereinander. Es werden beide `<title>` Elemente aus dem Quelldokument ausgegeben. Die Untertitel sind mit dem `@type`-Attribut “sub” gekennzeichnet.
Aus dem XML Dokument (Uebung_booklist.xml):

```
<bibl type="technology">
	<author>John Simpson</author>
	<title>XPath and XPointer</title>
	<title type="sub">Locating Content in XML Documents</title>
	<publisher>O'Reilly Media</publisher>
	<year>2002</year>
</bibl>
```

Will man nun Untertitel bei der Formatierung speziell berücksichtigen kann man `<xsl:choose>` verwenden. `<xsl:choose>` bildet den Rahmen für die Abfragen `<xsl:when>` (obligatorisch) und `<xsl:otherwise>`. Das Element `<xsl:when>` verlangt ein `@test` Attribut, welches eine Bedingung definiert. Trifft die definierte Bedingungen zu ist das Ergebnis der Booleschen Abfrage TRUE und die Anweisung, die im `<xsl:when>` Element definiert ist, wird ausgeführt.
Es können mehrere `<xsl:when>` aneinandergereiht werden. Optional kann auch ein `<xsl:otherwise>` definiert werden, welches ausgeführt wird, wenn alle `@test`-Abfragen nicht zutreffen.Ausgewählt wird diejenige Abfrage, deren Bedingung als erste zutrifft.
Mit der Syntax `<xsl:when test="title[@type='sub']">` wird gefragt, ob es ein `<title>` Element mit dem Attribut @type mit dem Attributwert 'sub' gibt. Tritt der Fall ein, werden die Befehle im `<xsl:when>` Element ausgeführt - zuerst der Titel ausgegeben, ein Halbgeviertstrich und dann der Untertitel. Tritt der Fall nicht ein, wird nur der Titel ausgegeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	version="2.0">
	<xsl:template match="bookList">
	<h1>Literaturliste</h1>
	<ul>
		<xsl:for-each select="//bibl">
		<xsl:choose>
			<xsl:when test="title[@type = 'sub']">
				<li><xsl:value-of select="title[not(@type=sub)]"/>
				<xsl:text> - </xsl:text>
				<xsl:value-of select="title[@type = 'sub']"/></li>
			</xsl:when>
			<xsl:otherwise>
				<li><xsl:value-of select="title"/></li>
			</xsl:otherwise>
		</xsl:choose>
		</xsl:for-each>
	</ul>
	</xsl:template>
</xsl:stylesheet>
```

Exkurs:
`<xsl:value-of select="title[not(@type=sub)]"/>`
Mit dem XPath Befehl `title[not(@type=sub)]` wird getestet, ob der Titel ein Untertitel ist. Die Funktion `not()` kehrt den Rückgabewert um: `TRUE` wird `FALSE`; `FALSE` wird `TRUE`. Genaugenommen wird getestet, ob etwas wahr ist und zurückgegeben, dass es falsch ist.

xsl:if

Eine andere Möglichkeit das Ergebnis zu erzielen wäre mit dem Element `<xsl:if>`.
Mit `<xsl:if>` wird die Ausführung einer Anweisung ebenfalls an eine Bedingung geknüpft. Nur wenn das Ergebnis der Abfrage den Booleschen Wert `TRUE` ergibt, also wenn die Bedingung erfüllt ist, werden weitere Operationen ausgeführt.
Die Syntax ist `<xsl:if test=”title[@type='sub']”>`

```
<li>
      <xsl:value-of select="author"/>
      <xsl:text>: </xsl:text>
      <xsl:value-of select="title[not(@type = 'sub')]"/>
      <xsl:if test="title[@type = 'sub']">
            <xsl:text> - </xsl:text>
            <xsl:value-of select="title[@type = 'sub']"/>
      </xsl:if>
</li>
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      exclude-result-prefixes="xs"
5      version="2.0">
6
7      <xsl:template match="bookList">
8          <h1>Literaturliste</h1>
9          <ul>
10             <xsl:for-each select="bibl">
11                 <li>
12                     <xsl:value-of select="author"/>
13                     <xsl:text>: </xsl:text>
14                     <xsl:value-of select="title[not(@type = 'sub')]"/>
15                     <xsl:if test="title[@type = 'sub']">
16                         <xsl:text> - </xsl:text>
17                         <xsl:value-of select="title[@type = 'sub']"/>
18                     </xsl:if>
19                 </li>
20             </xsl:for-each>
21         </ul>
22     </xsl:template>
23
24 </xsl:stylesheet>
```