

Transformationen mit XSLT



Prof. Dr. Christof Schöch

Modul Auszeichnungssprachen
MSc. Digital Humanities, Universität Trier





Überblick

1. Motivation: Wozu XSLT?
2. Ablauf der XSLT-Prozessierung
3. Bausteine eines XSLT-Stylsheets
4. Beispiel: Weinliste

(1) Motivation: Wozu XSLT?

Warum XSLT?

- CSS kann nur "stylen", aber nicht umorganisieren, sortieren, etc.
- Notwendig, wenn man XML-Dokumente nicht als Text, sondern als Daten sieht
- W3C: "Use CSS when you can, use XSL when you must."

Was ist XSLT?

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations
- Eine in XML geschriebene Sprache

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations
- Eine in XML geschriebene Sprache
- Steuert die Umwandlung eines XML-Dokuments in ein anderes Dokument

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations
- Eine in XML geschriebene Sprache
- Steuert die Umwandlung eines XML-Dokuments in ein anderes Dokument
- Kann auf beliebiges, wohlgeformtes XML angewandt werden

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations
- Eine in XML geschriebene Sprache
- Steuert die Umwandlung eines XML-Dokuments in ein anderes Dokument
- Kann auf beliebiges, wohlgeformtes XML angewandt werden
- Primär beschreibende Sprache (Templates), die aber auch Anweisungen ausführen kann

Was ist XSLT?

- XSLT: eXtensible Stylesheet Language for Transformations
- Eine in XML geschriebene Sprache
- Steuert die Umwandlung eines XML-Dokuments in ein anderes Dokument
- Kann auf beliebiges, wohlgeformtes XML angewandt werden
- Primär beschreibende Sprache (Templates), die aber auch Anweisungen ausführen kann
- Nutzt XPath, um die Teile des Dokuments zu adressieren

(2) Ablauf der XSLT-Prozessierung

XSLT-Processing (1)

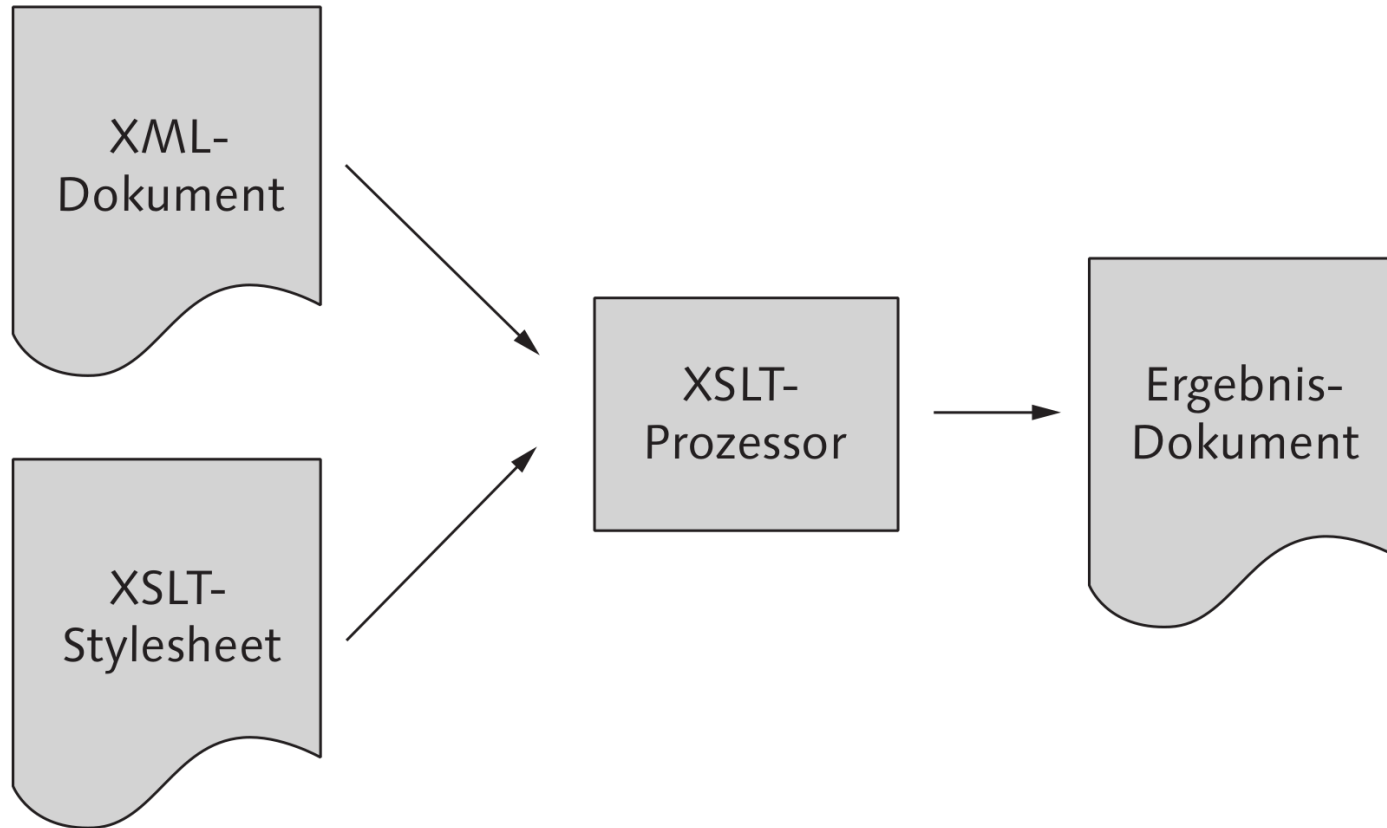


Abbildung 7.2 Generelles Schema der Transformation durch XSLT

(Quelle: Vonhoegen, *Einstieg in XML* (2007), Kapitel 7: "Umwandlung mit XSLT".)

Was passiert da genau?

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;
 2. Prüft für jeden Knoten, ob eine Regel im XSLT-Baum passt;

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;
 2. Prüft für jeden Knoten, ob eine Regel im XSLT-Baum passt;
 3. Wenn ja, wird die Regel angewandt; sonst wird weiter geprüft.

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;
 2. Prüft für jeden Knoten, ob eine Regel im XSLT-Baum passt;
 3. Wenn ja, wird die Regel angewandt; sonst wird weiter geprüft.
5. Dabei baut der XSLT-Processor einen neuen Baum für den Output

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;
 2. Prüft für jeden Knoten, ob eine Regel im XSLT-Baum passt;
 3. Wenn ja, wird die Regel angewandt; sonst wird weiter geprüft.
5. Dabei baut der XSLT-Processor einen neuen Baum für den Output
6. Ein XSLT "Serializer" transformiert den Baum in das gewünschte Format (HTML, SVG, XML, etc.)

Was passiert da genau?

1. XML Parser liest XML, konstruiert XML-Baum
2. XML Parser liest das XSLT Stylesheet, konstruiert XML-Baum
3. Beide XML-Bäume werden an den XSLT Processor geschickt
4. XSLT Prozessor arbeitet beide Dokumente ab:
 1. Er geht in Dokumentreihenfolge durch den XML-Baum;
 2. Prüft für jeden Knoten, ob eine Regel im XSLT-Baum passt;
 3. Wenn ja, wird die Regel angewandt; sonst wird weiter geprüft.
5. Dabei baut der XSLT-Processor einen neuen Baum für den Output
6. Ein XSLT "Serializer" transformiert den Baum in das gewünschte Format (HTML, SVG, XML, etc.)
7. Das neue Dokument wird abgespeichert oder im Browser angezeigt

XSLT-Processing (2)

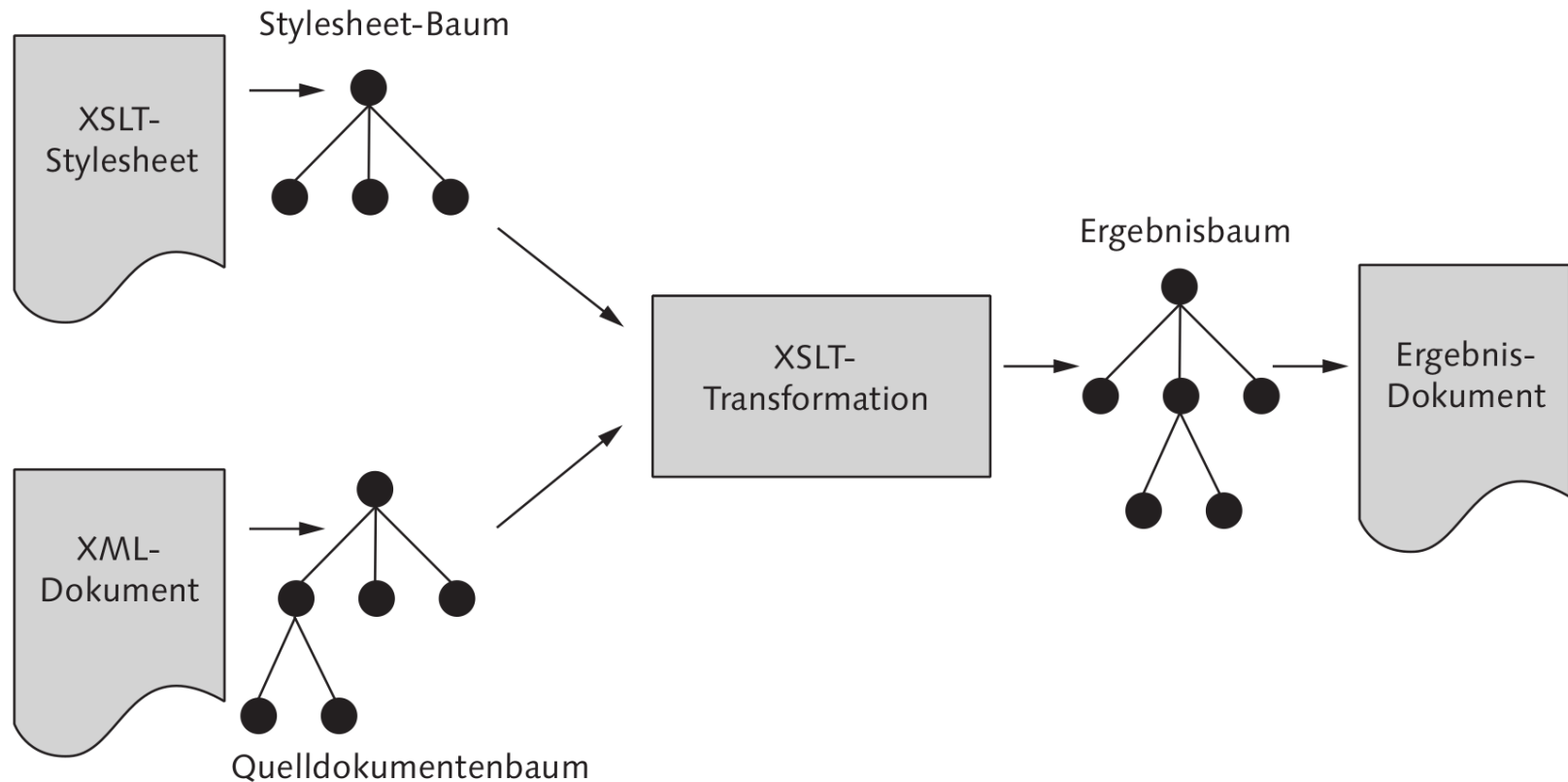


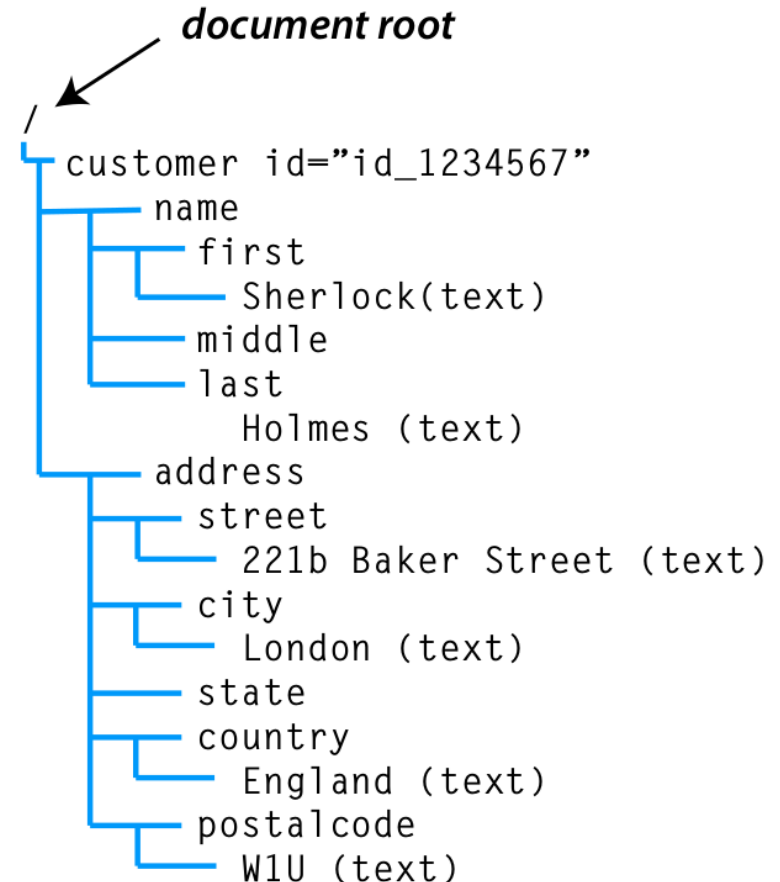
Abbildung 7.8 Schema der Transformation im Detail

(Quelle: Vonhoegen, *Einstieg in XML* (2007), Kapitel 7: "Umwandlung mit XSLT".)

XML-Baum und Dokumentreihenfolge

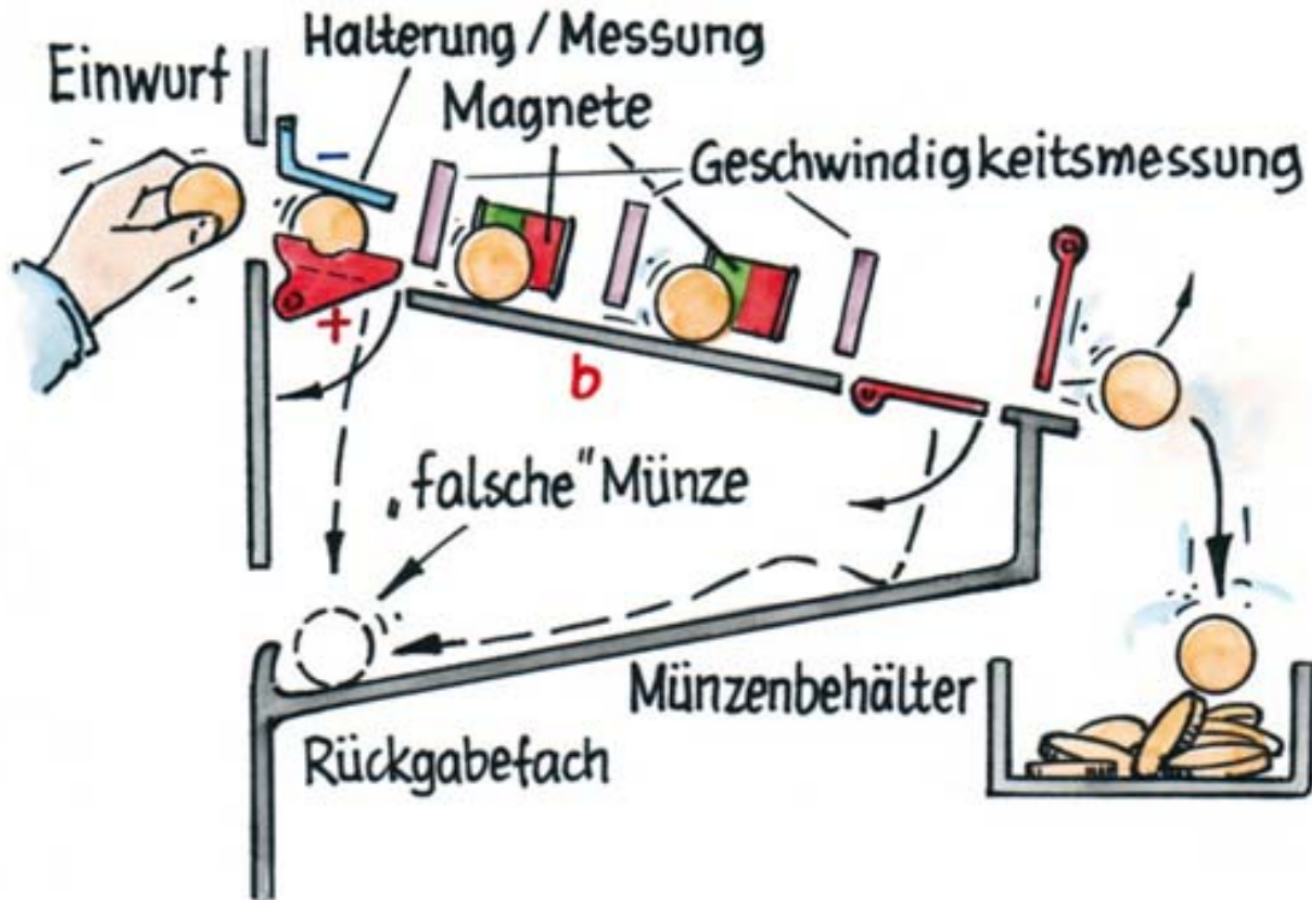
```
<?xml version='1.0' encoding='UTF-8'?>  
<?xml-stylesheet type="text/xsl" href="bib1.xsl"?>
```

```
<customer id="id_1234567">  
  <name>  
    <first>Sherlock</first>  
    <middle></middle>  
    <last>Holmes</last>  
  </name>  
  <address>  
    <street>221b Baker Street</street>  
    <city>London</city>  
    <state></state>  
    <country>England</country>  
    <postalcode>W1U</postalcode>  
  </address>  
</customer>
```



(Quelle: Kelly, *XSLT Jumpstarter* (2015), Chapter 1: "Introducing XSLT")

XSLT-Prozessor als "Münzprüfer"



(Martin Apolin, "Die Arbeitsweise der Physik", 2010, www.oebv.at.)

(3) Bausteine eines XSLT- Stylesheets

Grundgerüst

E07-xslt-struktur.xsl

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3   xmlns="http://www.w3.org/1999/xhtml"
4   version="1.0">
5
6   <xsl:template match="/">
7     <!-- weitere Anweisungen -->
8   </xsl:template>
9
10  <!-- weitere Templates -->
11
12 </xsl:stylesheet>
```

- XML Declaration
- Stylesheet Declaration
- Eines oder mehrere Templates

Das `<xsl:template>`-Tag

- Grundlegender Baustein
- Beinhaltet mehrere Dinge
 - das `match`-Attribut bestimmt, ob/wann das Template eingesetzt wird
 - Weitere Prozesse können ausgelöst werden: Variablen, Berechnungen, etc.
 - auch "literale" Inhalte können vorkommen (bspw. HTML-Tags)
 - Instruktionen, weitere Templates aufzurufen (`xsl:apply-templates`)

Das `match=""`-Attribut

- Befindet sich innerhalb des `xsl:template` Tags
- Evaluiert den Wert relativ zum aktuellen Kontextknoten
- Gefunden werden können:
 - Attribute des Kontextknotens
 - Text-Knoten des Kontextknotens
 - Direkte Kind-Knoten des Kontextknotens
- Mehr Flexibilität durch:
 - Einsatz von XPath als Wert von `match`
 - Nicht an die Dokumentreihenfolge und den aktuellen Kontext gebunden

Das `<xsl:apply-templates>`-Tag

- Fordert den Prozessor auf:
 - weiter im XML-Dokument Knoten aufzurufen
 - und nach passenden XSLT-Templates zu suchen
- `select`-Attribut
 - zur Fokussierung dieses Prozesses
 - funktioniert wie ein Filter
 - Wert kann ein Knoten-Namen oder ein XPath sein
- `mode`-Attribut
 - verbindet `xsl:template`-Tags mit `xsl:apply-templates`-Tags
 - so kann ein XML-Knoten mehrfach aktiviert werden
 - bspw. Überschriften im Text und im Inhaltsverzeichnis
- aktiviert außerdem die "built-in templates" von XSLT

Das `<xsl:for-each>`-Tag

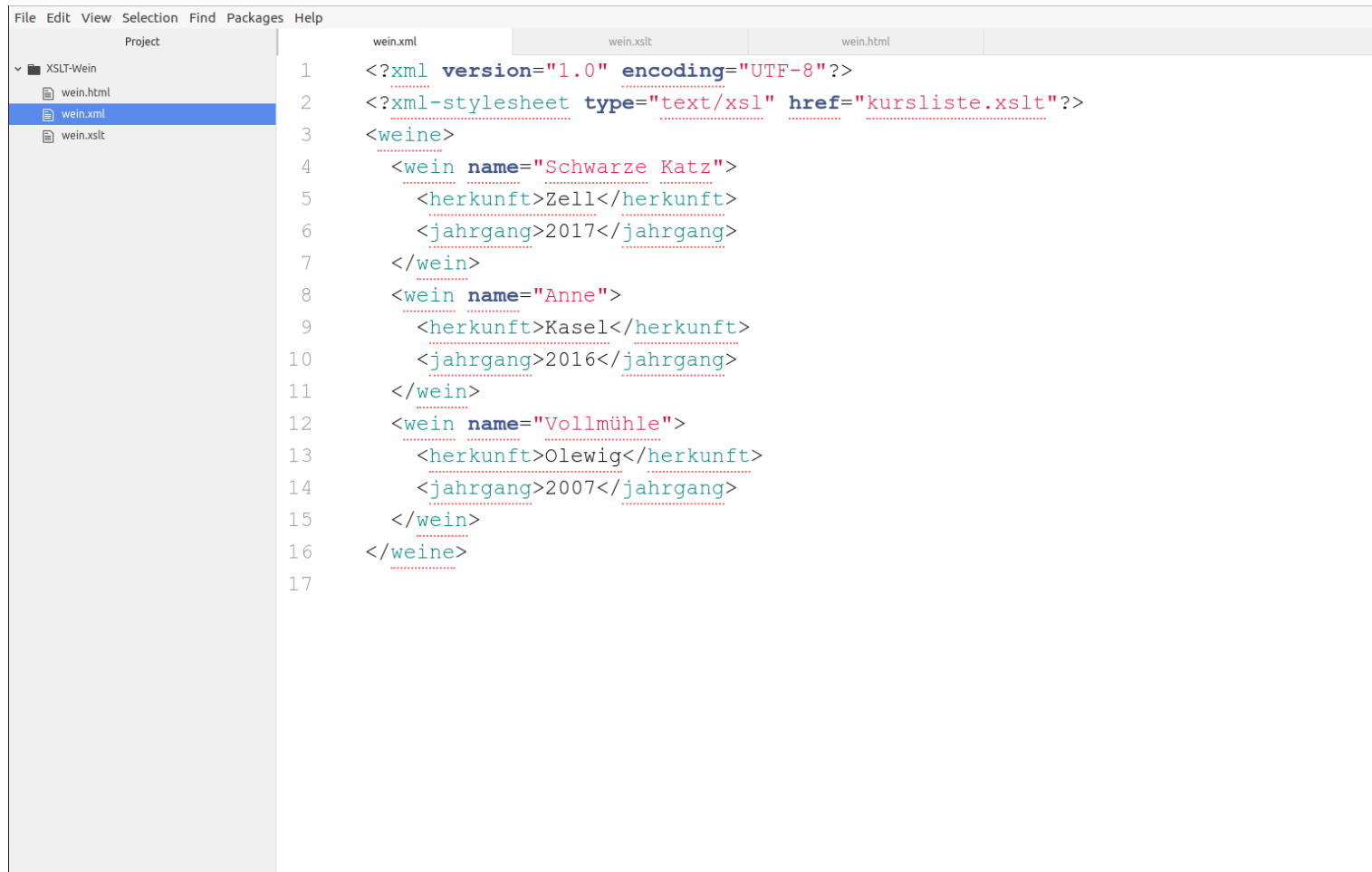
- Der zentrale Mechanismus für Schleifen in XSLT
- Genauer: keine Schleife (mit Zähler), sondern eine Iteration
- Der XPath auf dem `select`-Attribut bestimmt den Anwendungsbereich
- Auf jedem der Treffer-Knoten werden dann Anweisungen ausgeführt

Das `xsl:value-of`-Tag

- dient der Datenübernahme aus dem XML-Dokument
- `select`-Attribut
 - bestimmt, was genau übernommen werden soll
 - XPath: Elemente, Attribute, Werte, Textinhalte

Beispiel: Weinliste

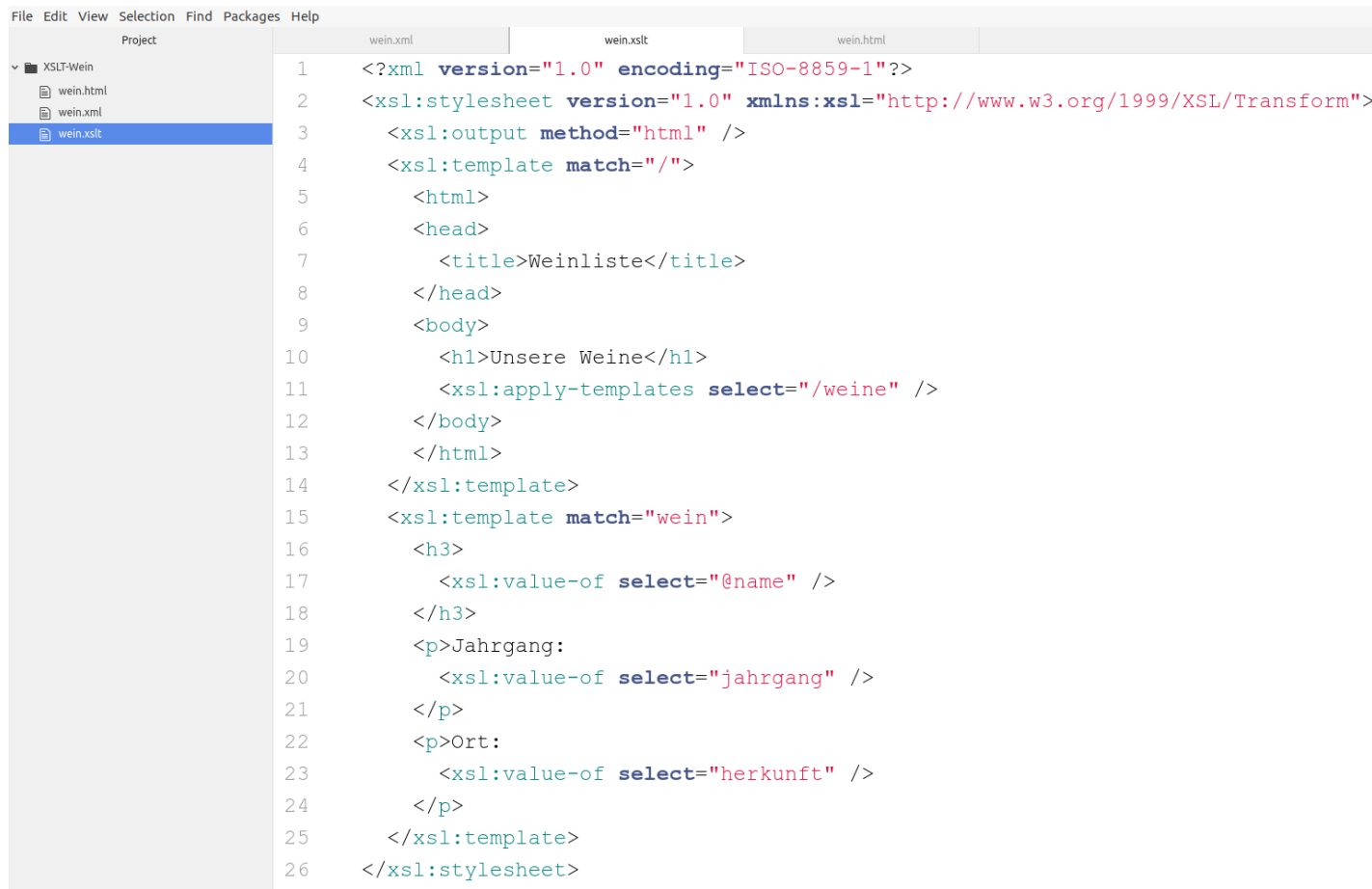
XML-Dokument (Atom)



The screenshot shows an IDE interface with a project sidebar on the left and a code editor on the right. The sidebar, titled 'Project', shows a folder 'XSLT-Wein' containing three files: 'wein.html', 'wein.xml' (selected), and 'wein.xslt'. The code editor displays the content of 'wein.xml' with line numbers 1 through 17. The XML document is an Atom feed for wine data, featuring a root element 'weine' containing three 'wein' elements. Each 'wein' element has attributes 'name' and 'herkunft', and a child element 'jahrgang'.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="kursliste.xslt"?>
3 <weine>
4   <wein name="Schwarze Katz">
5     <herkunft>Zell</herkunft>
6     <jahrgang>2017</jahrgang>
7   </wein>
8   <wein name="Anne">
9     <herkunft>Kasel</herkunft>
10    <jahrgang>2016</jahrgang>
11  </wein>
12  <wein name="Vollmühle">
13    <herkunft>Olewig</herkunft>
14    <jahrgang>2007</jahrgang>
15  </wein>
16 </weine>
17
```

XSLT-Stylesheet (Atom)

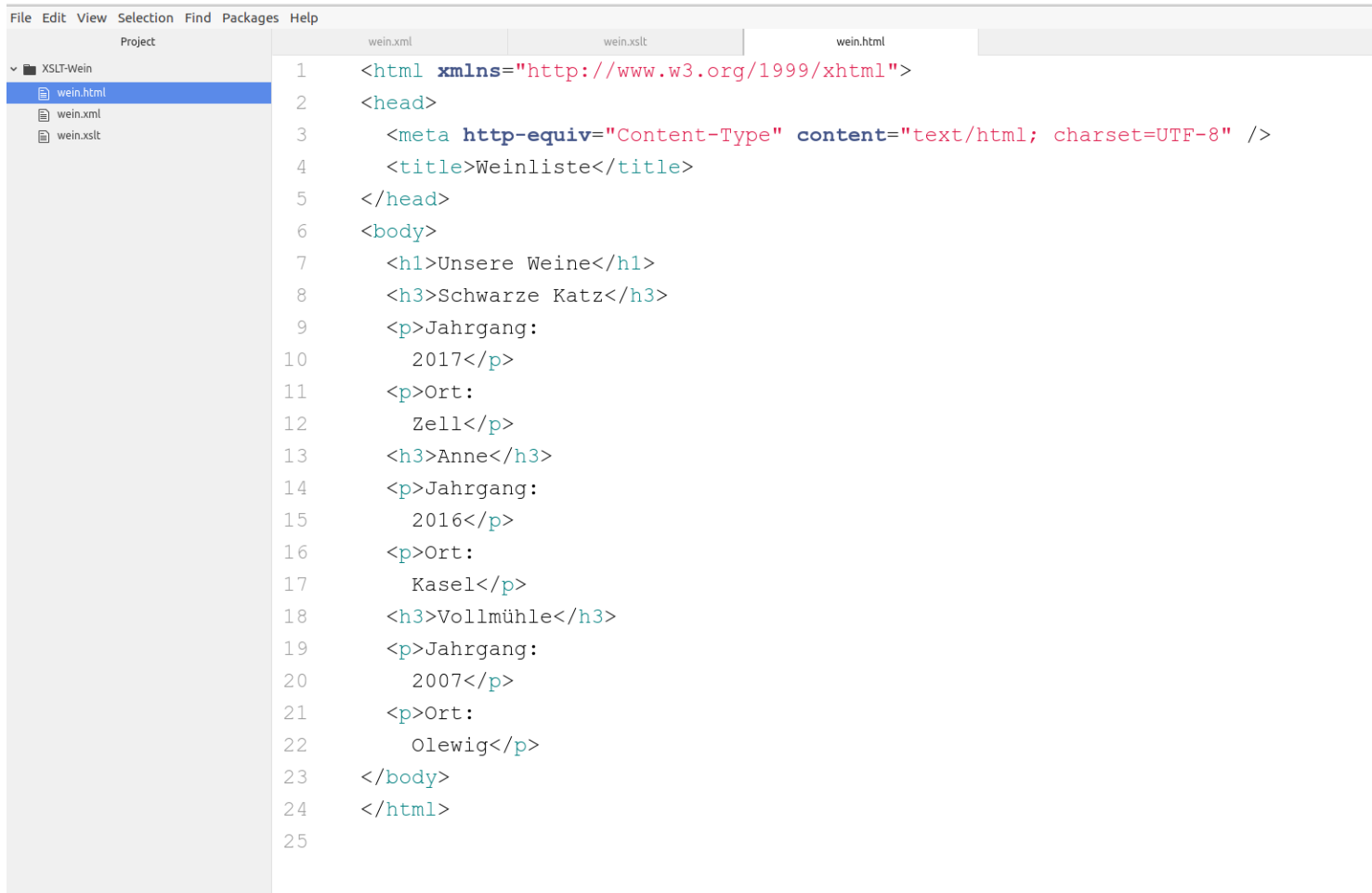


The screenshot shows an IDE window with a project named 'XSLT-Wein'. The project contains three files: 'wein.html', 'wein.xml', and 'wein.xslt'. The 'wein.xslt' file is selected and its content is displayed in the editor. The code is an XSLT stylesheet that transforms an XML document into HTML. It starts with an XML declaration, followed by an XSL stylesheet declaration with version '1.0' and namespace 'http://www.w3.org/1999/XSL/Transform'. The output method is set to 'html'. A template with match='/' is defined, which outputs an HTML document with a title 'Weinliste', a head section, and a body section. The body section contains a heading 'Unsere Weine' and a template for each 'weine' element. This template outputs a heading 'Jahrgang:' followed by the value of the 'jahrgang' attribute, and another heading 'Ort:' followed by the value of the 'herkunft' attribute.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:output method="html" />
4   <xsl:template match="/">
5     <html>
6     <head>
7       <title>Weinliste</title>
8     </head>
9     <body>
10      <h1>Unsere Weine</h1>
11      <xsl:apply-templates select="/weine" />
12    </body>
13  </html>
14 </xsl:template>
15 <xsl:template match="wein">
16   <h3>
17     <xsl:value-of select="@name" />
18   </h3>
19   <p>Jahrgang:
20     <xsl:value-of select="jahrgang" />
21   </p>
22   <p>Ort:
23     <xsl:value-of select="herkunft" />
24   </p>
25 </xsl:template>
26 </xsl:stylesheet>
```

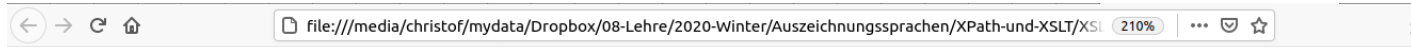
- Transformation: mit Plugin "XSLTransform"
- Strg+P, "XSLT", Dateiauswahl, Enter

HTML-Output (Atom)



```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4   <title>Weinliste</title>
5 </head>
6 <body>
7   <h1>Unsere Weine</h1>
8   <h3>Schwarze Katz</h3>
9   <p>Jahrgang:
10     2017</p>
11   <p>Ort:
12     Zell</p>
13   <h3>Anne</h3>
14   <p>Jahrgang:
15     2016</p>
16   <p>Ort:
17     Kasel</p>
18   <h3>Vollmühle</h3>
19   <p>Jahrgang:
20     2007</p>
21   <p>Ort:
22     Olewig</p>
23 </body>
24 </html>
25
```

HTML-Output (Browser)



Unsere Weine

Schwarze Katz

Jahrgang: 2017

Ort: Zell

Anne

Jahrgang: 2016

Ort: Kasel

Vollmühle

Jahrgang: 2007

Ort: Olewig

Abschluss

Lektürehinweise

Grundlagen / Referenz

- David James Kelly: "Chapter 1: Introducing XSLT" und "Chapter 2: XSLT in Action", in: *XSLT Jumpstarter*. Raleigh, NC: Peloria Press, 2015, S. 1-29.

Weitere Empfehlungen zur Vertiefung

- Helmut Vonhoegen: "Umwandlungen mit XSLT", in: *Einstieg in XML: Grundlagen, Praxis, Referenzen*. 4. Auflage. Bonn: Galileo Press, 2007, S. 241-327.
- Michael Kay: *XSLT 2.0 and XPath 2.0 Programmer's Reference*. 4th Edition. Indianapolis IN: Wiley, 2008. (1316 Seiten!)

Danke!

Lizenz: [Creative Commons Attribution \(CC BY\)](#), 2020.
