XSLT

XSLT – Was ist das?

XSLT (Extensible Stylesheet Language Transformation) ist eine Transformationssprache für XML-Dokumente.

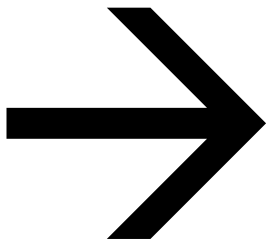
XSLT – Was ist das?

W3C-Recommendation:
http://www.w3.org/Style/XSL/

W3C-Standard seit 1999, aktuell: XSLT 3.0

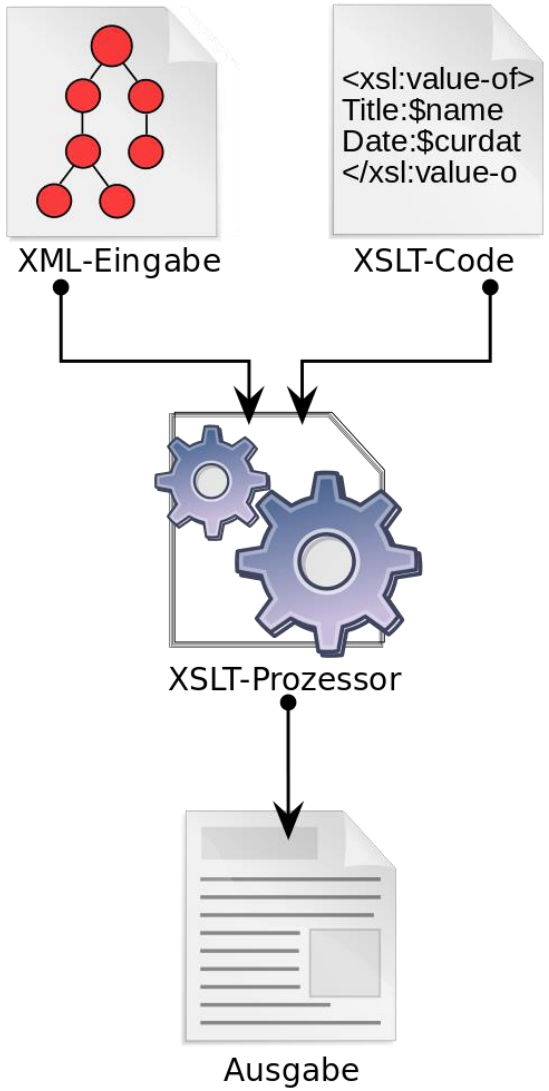
Beispiel für eine Transformation

```
<persons>
  <person>
    <name>Alice</name>
    <place>New York</place>
  </person>
  <person>
    <name>Bob</name>
    <place>London</place>
  </person>
  <person>
    <name>Caroline</name>
    <place>Berlin</place>
  </person>
</persons>
```



```
<ul>
  <li>Alice (New York)</li>
  <li>Bob (London)</li>
  <li>Caroline (Berlin)</li>
</ul>
```

Wie funktioniert eine XSLT-Transformation?



Beispiel für eine XSLT-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <ul>
      <xsl:for-each select="//person">
        <li><xsl:value-of select="name"/> (<xsl:value-of select="place"/>)</li>
      </xsl:for-each>
    </ul>
  </xsl:template>
</xsl:stylesheet>
```

Beispiel für eine XSLT-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <ul>
      <xsl:for-each select="//person">
        <li><xsl:value-of select="name"/> (<xsl:value-of select="place"/>)</li>
      </xsl:for-each>
    </ul>
  </xsl:template>
</xsl:stylesheet>
```

Verschiedene Technologien und Formate in XSLT

In einem XSLT-Dokument kommen also an Sprachen und XML-basierten Technologien mindestens vor:

- XML als Basis für XSLT
- XSLT selbst (XML-basiert)
- XPath
- Das Format des Quell-Dokuments (XML)
- Das Format des Ziel-Dokuments (z. B. XML, HTML, XHTML oder reiner Text)
- Zur Unterscheidung der einzelnen verwendeten Sprachen werden XML-Namespaces verwendet.

Das Wurzel-Element <xsl:stylesheet>

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
exclude-result-prefixes="xs">
  <xsl:template match="/">
    <ul>
      <xsl:for-each select="//person">
       <li><xsl:value-of select="name"/></li>
      </xsl:for-each>
    </ul>
  </xsl:template>
</xsl:stylesheet>
```

XSLT-Stylesheet in der Quelldatei referenzieren

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

<xsl:output>

Das Top-Level-Element <xsl:output> gibt an, welche Art der Ausgabe der Prozessor generieren soll. Dabei kommen drei Ausgabeformate in Frage: XML, HTML sowie Text und in XSLT 2.0 noch XHTML. Diese werden als Werte des Attributes method angegeben. Standardausgabe von XSLT ist XML, es sei denn, es findet sich ein HTML-Tag im Dokument, dann wird HTML 4.0 generiert.

Quelle: https://www.data2type.de/xml-xslt-xslfo/xslt/xslt-einfuehrung/output-element/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" version="4.01" encoding="UTF-8"/>
</xsl:stylesheet>
```

<xsl:template>

Mit dem <xsl:template>-Element kann man Templates bauen.

Das **match**-Attribut wird benutzt, um ein Template einem bestimmten XML-Knoten zuzuordnen. Der Wert des match-Attributs ist ein XPath-Ausdruck.

Das match-Attribut kann auch benutzt warden, um für das ganze XML-Dokument ein Template zu definieren, indem man den XPath-Ausdruck "/" für das ganze Dokument benutzt.

<xsl:template>

```
<xsl:template match="/">
  <ul>
    <xsl:for-each select="//person">
     <li><xsl:value-of select="name"/></li>
    </xsl:for-each>
  </ul>
</xsl:template>
```

Weitere Infos: https://www.data2type.de/xml-xslt-xslfo/xslt/xslt2/start-xslt-xpath/grundstruktur-template/

<xsl:value-of>

Das Element <xsl:value-of> gibt den Wert des ausgewählten Knoten zurück.

```
<p><xsl:value-of select="catalog/cd/title"/></p>
```

<xsl:for-each>

Das Element <xsl:for-each> kann benutzt warden, um jedes XML-Element eines angegebenen Node-Set zu verarbeiten.

```
<xsl:for-each select="catalog/cd">
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
```

<xsl:sort>

Um Elemente zu sortieren, kann ein Tag for-each mit einem sort kombiniert werden. Die Schleife läuft dann nicht in der Reihenfolge der Knoten des Originaldokuments, sondern in alphabetischer oder numerischer Reihenfolge.

```
<xsl:for-each select="catalog/cd">
  <xsl:sort select="title" order="ascending" />

  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
```

<xsl:apply-templates>

<xsl:apply-templates> transformiert die Kindelemente des aktuellen Elements mittels sämtlicher dafür anwendbarer Regeln (Templates).

Fügt man ein select-Attribut hinzu, verarbeitet <xsl:apply-templates> nur das Kind-Element, das auf den Wert des select-Attributs passt.

Ferner kann das select-Attribut auch benutzt werden, um die Reihenfolge zu bestimmten, in welcher die Kind-Knoten verarbeitet werden sollen.

<xsl:apply-templates>

Beispiel:

https://www.w3schools.com/xml/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog_apply

<xsl:variable>

Mit dem Element <xsl:variable> kann man locale oder globale Variablen deklarieren. Die Variable ist global, wenn sie als Top-Level-Element positioniert ist, und local, wenn sie innerhalb eines Templates positioniert ist.

Der Wert einer Variablen kann nicht nachträglich verändert werden.

Der Wert einer Variablen kann entweder per **select**-Attribut, oder per Content des <xsl:variable>-Elements!

<xsl:variable>

```
<xsl:variable name="color" select="red" />

<xsl:variable name="header">
  <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
  </tr>
</xsl:variable>

<xsl:copy-of select="$header" />
```

<xsl:copy>

Das Element <xsl:copy> erstellt eine Kopie des aktuellen Knotens.

Achtung: Kindknoten und Attribute des aktuellen Knotens werden nicht automatisch mit kopiert!

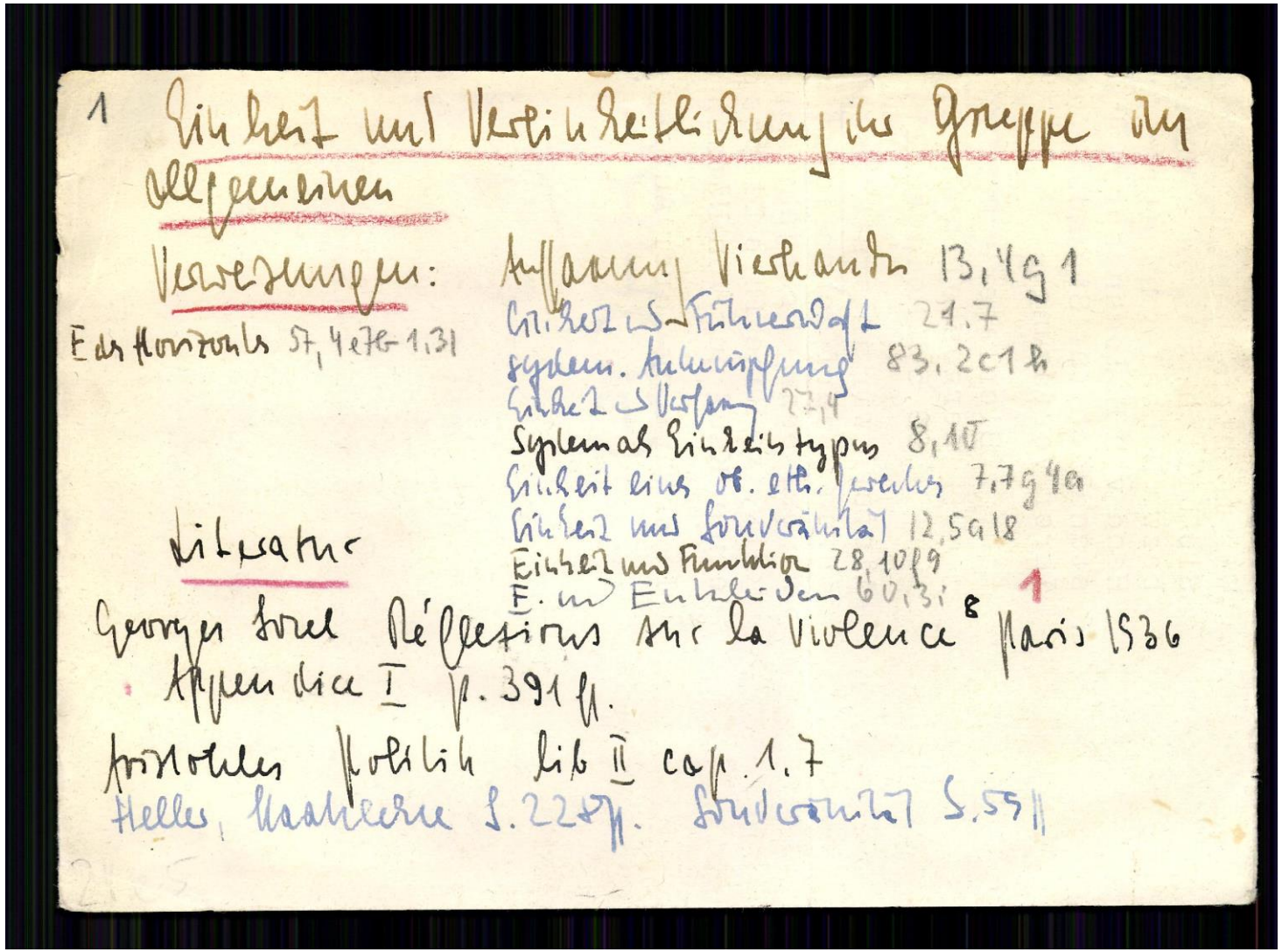
```
<xsl:template match="div">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
```

<xsl:copy-of>

Das Element <xsl:copy> erstellt eine Kopie des aktuellen Knotens mit Kindknoten und Attributen des aktuellen Knotens.

```
<xsl:template match="div">
  <xsl:copy-of />
</xsl:template>
```

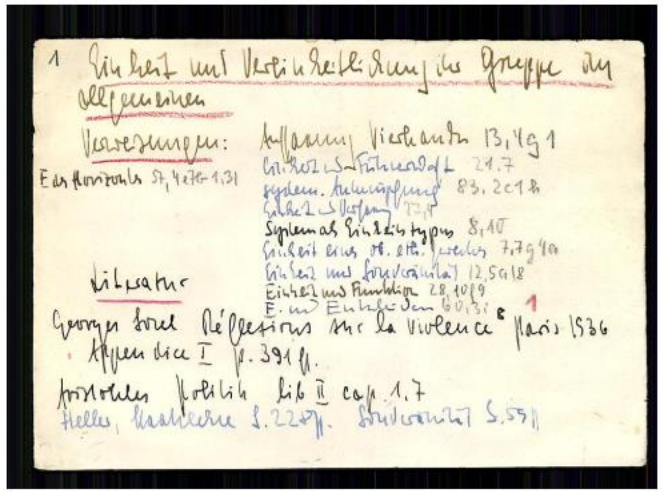
Beispiel: TEI nach HTML



Suche

Suchbegriff

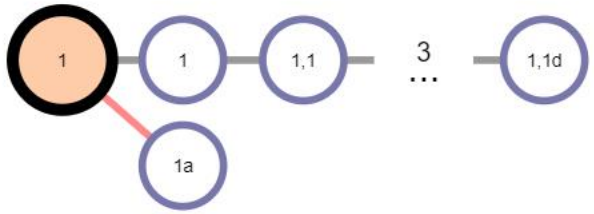
ZK I: Zettel 1



1 Einheit und Vereinheitlichung der Gruppe im allgemeinen

Verweisungen: Auffassung Vierkandt13,4g1
Einheit und Führerschaft 21,7
E des Horizontes 57,4e7b1,31
system. Anknüpfung 83,2c1h
Einheit und Verfassung 27,4
System als Einheitstypus 8,10
Einheit eines ob. eth. Zweckes 7,7g4a
Einheit und Souveränität 12,5a18
Einheit und Funktion 28,10f9
E. und Entscheiden 60,3i 1

Literatur
Georges Sorel, Réflexions sur la violence[8], Paris 1936,
Appendice I, p. 391ff.
Aristoteles, Politik, lib II cap. 1,7
Heller, Staatslehre, S. 228ff.; Souveränität, S. 59ff.



Übung

Was macht diese Transformation?

```
<?xml version="1.0" ?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Übung

Erstelle eine Transformation, die rekursiv alle <movie>-Elemente in <film>-Elemente umwandelt.

Übung

```
<?xml version="1.0" ?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

  <!-- Rename para to p -->
  <xsl:template match="p">
     <paragraph>
       <xsl:apply-templates select="@* | node()" />
     </paragraph>
  </xsl:template>

</xsl:stylesheet>
```

Tipp zum Abschluss

Bei speziellen Probleme hilft oftmals googeln!