



# Wege zur Anzeige

XSL – die „eXtensible Stylesheet Language“



# Einführung

- Warum XML? Warum TEI?
  - Trennung von Form und Inhalt
  - Eine Quelle, viele verschiedene Ausgabeformate
- Standardstylesheets der TEI im Oxygen
  - Zufrieden?
  - Importieren und Überschreiben
- Eigene Stylesheets entwerfen

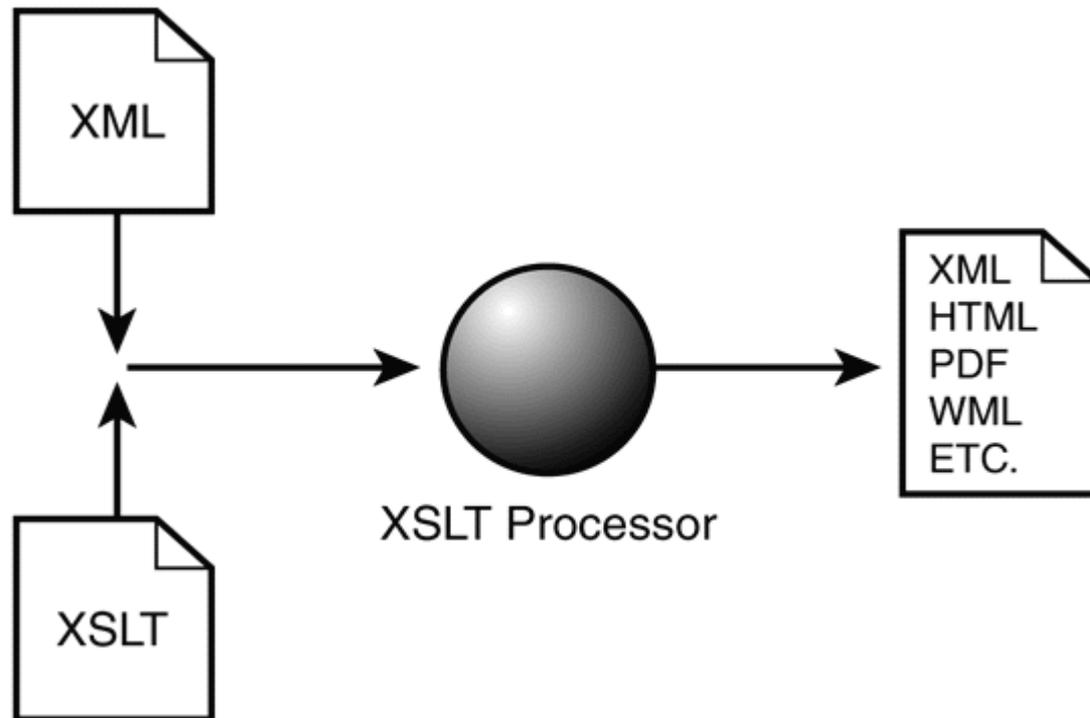


# Was ist XSL?

- Abkürzung für „eXtensible Stylesheet Language“
- Mehrere Komponenten:
  - XSLT → Transformations
  - XSL-FO → Formatting Objects
  - XPath
  - (XML-Schema)
- Ausgabeformate: HTML, PDF, RTF, WORD, TUSTEP, LaTeX
  - Aber auch XML: KML, WML, SVG, SMIL, DocBook, TEI

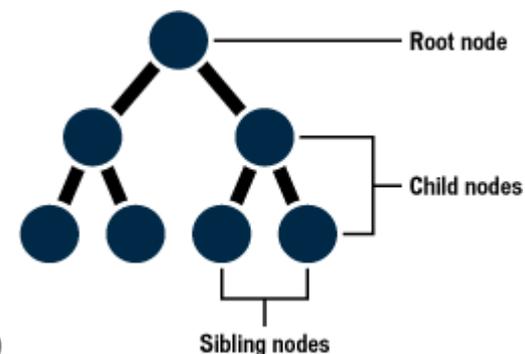


# XSLT Verarbeitung



# Wie funktioniert das genau?

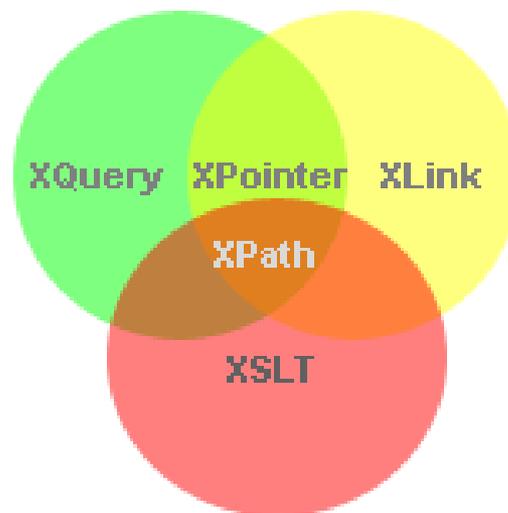
- Der Prozessor traversiert den Baum!
  - Er startet bei der Wurzel
  - Geht Schritt für Schritt vor
  - Und von links nach rechts (oder oben nach unten)
  - Und führt die ihm aufgetragenen Aufgaben (Transformationen) durch.
  
- Woher weiß der Prozessor, wann er was machen soll?
  - **XPath**
  - `<xsl:apply-templates/>`





# XPath Einführung

- Empfehlung des W3C
- Version 2.0 vom Dezember 2010
- Technologie zur Adressierung von Knoten
- Enthält nützliche Funktionen
- Achsenbasierte Navigation durch die Hierarchieebenen



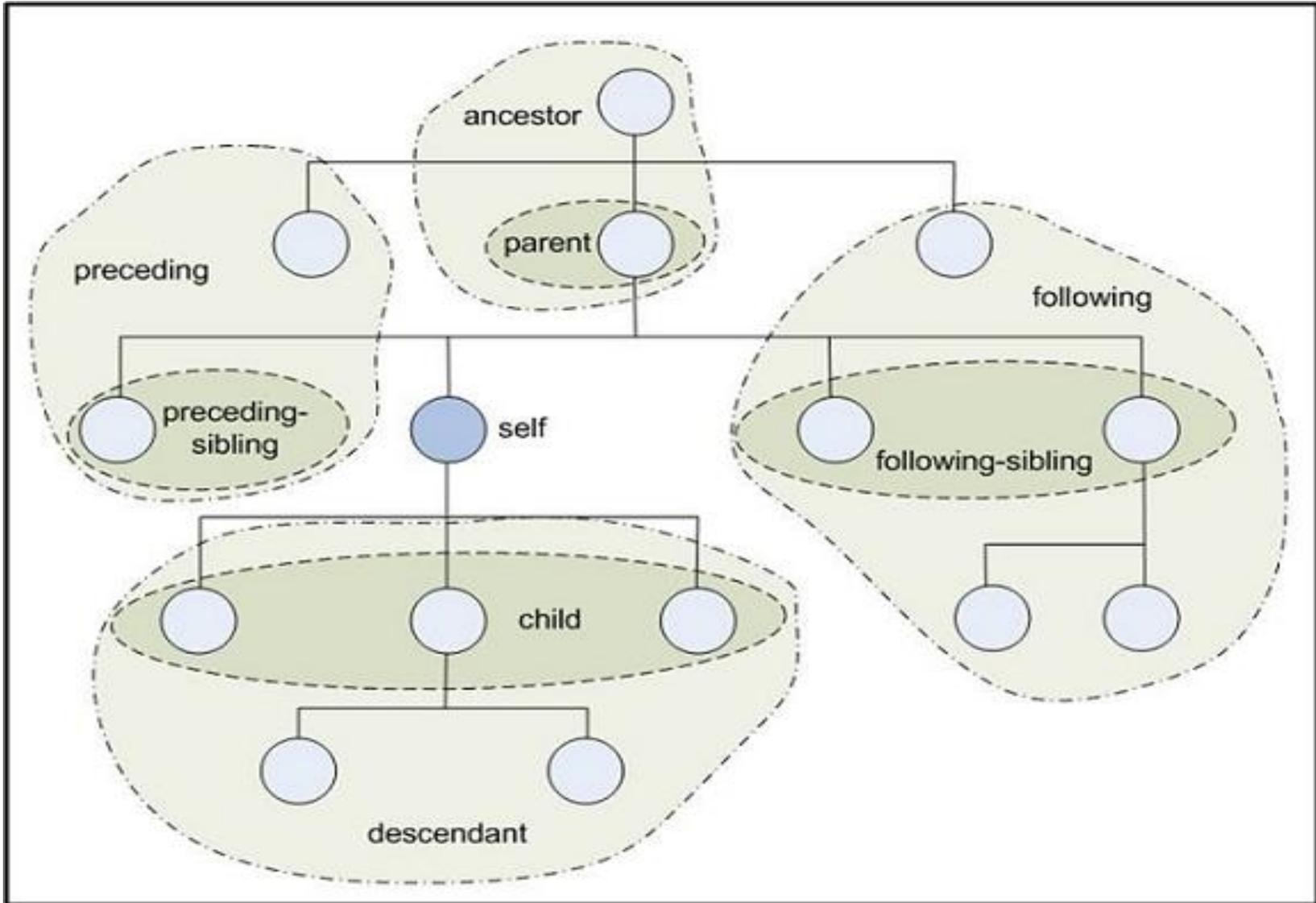
**Ohne XPath ist eine Verarbeitung und Abfrage von XML-Dokumenten unmöglich!**



# XPath Knotentypen

## 7 Knotentypen

- Wurzel- und Elementknoten: `<person>...</person>`
- Text- und Attributknoten: `<el f=„attribut“>text</el>`
- Namensraumknoten: `<tei:lb/>`
- Processing-Instruction-Knoten `<?...?>`
- `<!--` Kommentarknoten `-->`





# XPath Notation

- Nach dem Prinzip: **achse::knotentest([prädikat])**
  - /
  - /child::persName
  - /descendant::persName
  - /parent::\*
  - /child::person/attribute::role
  - /self::TEI/child::\*

Übung!



# XPath verkürzte Notation

- Nach dem Prinzip: **achse::knotentest([prädikat])**
  - / → /
  - /child::persName → /persName
  - /descendant::persName → //persName
  - /parent::\* → ..
  - /child::person/attribute::role → /person/@role
  - /self::TEI/child::\* → ./\*

Übung!



# XPath Prädikate

- Bedingungen, die Submengen definieren
  - `//person[@role = "member"]`
  - `/persName[surname = "müller"]`
  - `/persName[surname != "müller"]`
  - `//person[1]`
  - `//person[last()]/persName`
  - `//person[position() > 5]`

Übung!



# XPath Funktionen

- Dienen zur weiteren Verarbeitung und Adressierung von Knotenmengen
- 4 Gruppen von Funktionen
  - **Knotenmengenfunktionen**
  - **Zeichenkettenfunktionen**
  - Logische (boolsche) Funktionen
  - Numerische Funktionen



# XPath Funktionen

- Knotenmengen
  - last(), position(), current(), count(), ...
- Zeichenketten
  - concat()
  - contains()
  - starts-with()
  - ends-with()
  - substring()
  - translate()
  - matches()

Übung!

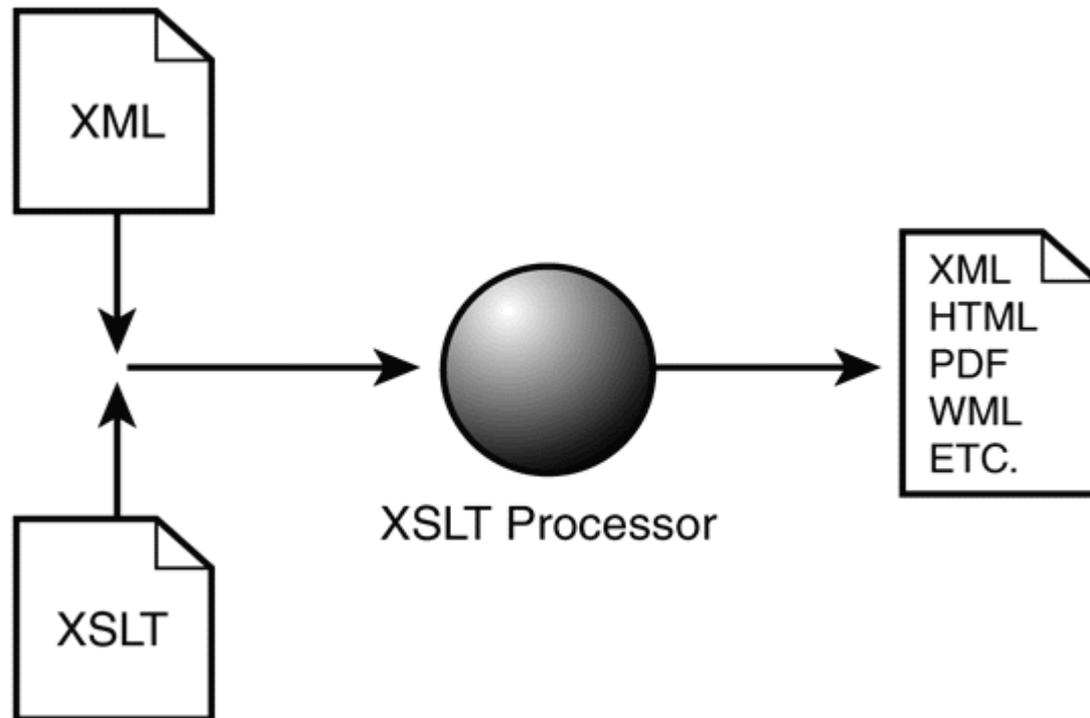


# XSLT

- W3C-Standard vom Januar 2007 (XSLT 2.0)
- „eXtensible Stylesheet Language“ for Transformations
- Sprache zur Spezifikation von Transformationen von XML-Dokumenten
- Dient zur Umwandlung eines XML-Dokumentes in ein Dokument einer anderen Form oder XML
- XML-Syntax
- Musterbasiert



# XSLT Verarbeitung





# XSLT Beispiel

- XML:

```
<message>Hallo Wien!</message>
```

- XSLT:

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html><body>
      <h1><xsl:value-of select="message"/></h1>
    </body></html>
  </xsl:template>
</xsl:stylesheet>
```



# Grundgerüst

- `<xsl:template match="//*">`  
     ...Anweisungen...  
`</xsl:template>`

- Schablone als Anweisung, an welcher Stelle im XML-Dokument (match="XPath") etwas gemacht werden soll
- Wichtigstes Toplevel-Element in XSLT

**Alle folgenden Elemente sind immer Kinder von  
`<xsl:template/>`!**



# Grundgerüst II

- `<xsl:apply-templates select="XPath"/>`
  - Anweisung, dass die Verarbeitung weiterer Templates fortgeführt werden soll.
  - Optional kann die zu verarbeitende **Knotenmenge** über das Attribut „select“ per XPath bestimmt werden.

Beispiel:

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



# Ausgabe von Text

- `<xsl:value-of select="XPath" />`
  - Wählt den **Wert** eines XPath-Ausdrucks, also **Text** eines Elements oder Attributs und fügt diesen in die Ausgabe ein.
  - `<h1> <xsl:value-of select="message" /> </h1>`
- `<xsl:text>` Schreibe diesen Text... `<xsl:text>`
- Wer findet den Fehler?



# Kopieren

- `<xsl:copy/>`
  - Kopiert den aktuellen Knoten inklusive Namespace
  - Kopiert keine Attribute, Inhalte, Kindknoten
- `<xsl:copy-of select="XPath"/>`
  - Kopiert den ausgewählten Knoten inklusive aller Kindknoten und Attribute



# Schleifen

- `<xsl:for-each select="XPath">`  
    ...  
    `</xsl:for-each>`
- Beispiel: Wähle jedes Buch (`//book`) und generiere eine Aufzählung (`<ul>`) der Titel (`title`):  
    `<ul>`  
        `<xsl:for-each select="//book">`  
            `<li> <xsl:value-of select="title"/> </li>`  
        `</xsl:for-each>`  
    `</ul>`



# Bedingungen

- `<xsl:if test="Bedingung">...</xsl:if>`
  - Führt Anweisungen nur aus, wenn die Bedingung im Attribut „test“ erfüllt ist

Beispiel:

```
<xsl:for-each select="//book">  
  <xsl:if test="author='Max Frisch'">  
    <li>  
      <xsl:value-of select="title"/>  
    </li>  
  </xsl:if>  
</xsl:for-each>
```



# Sortierungen

- `<xsl:sort select="Sortierkriterium"/>`
  - Sortiert die Elemente einer Schleife (nach dem ausgewählten Kriterium)
  - Wird immer innerhalb von `<xsl:for-each/>` verwendet

Beispiel:

`<ul>`

`<xsl:for-each select="//book">`

`<xsl:sort select="author"/>`

`<li> <xsl:value-of select="title"/>`

`<xsl:text> von </xsl:text>`

`<xsl:value-of select="author"/> </li>`

`</xsl:for-each>`



# Selber machen!?

- Ein leicht verständliches XPath-Tutorial: [Link](#)
- Zugriff auf XML-Dokumente (die XPath-Achsen visualisiert): [Link](#)
- Ein einfaches XSLT-Tutorial, welches u.a. gut den Unterschied zwischen `xsl:value-of` und `xsl:apply-templates` deutlich macht: [Link](#)
- Die dazugehörige XSLT-Referenz: [Link](#)
- Weitere, nützliche XSL-Tutorials: [Link](#)
- I|D|E-XML-Kurzreferenz: [Link](#)