CLUG

XSLT

Chris Hübsch

Zusammenfassung

Immer mehr Daten werden in XML-Formaten gespeichert. Eine Kernaufgabe bei der Verarbeitung dieser Daten ist die Umwandlung in andere Formate (ASCII oder XML). Mit XSLT wurde eine Sprache für genau diese Aufgabe entwickelt.

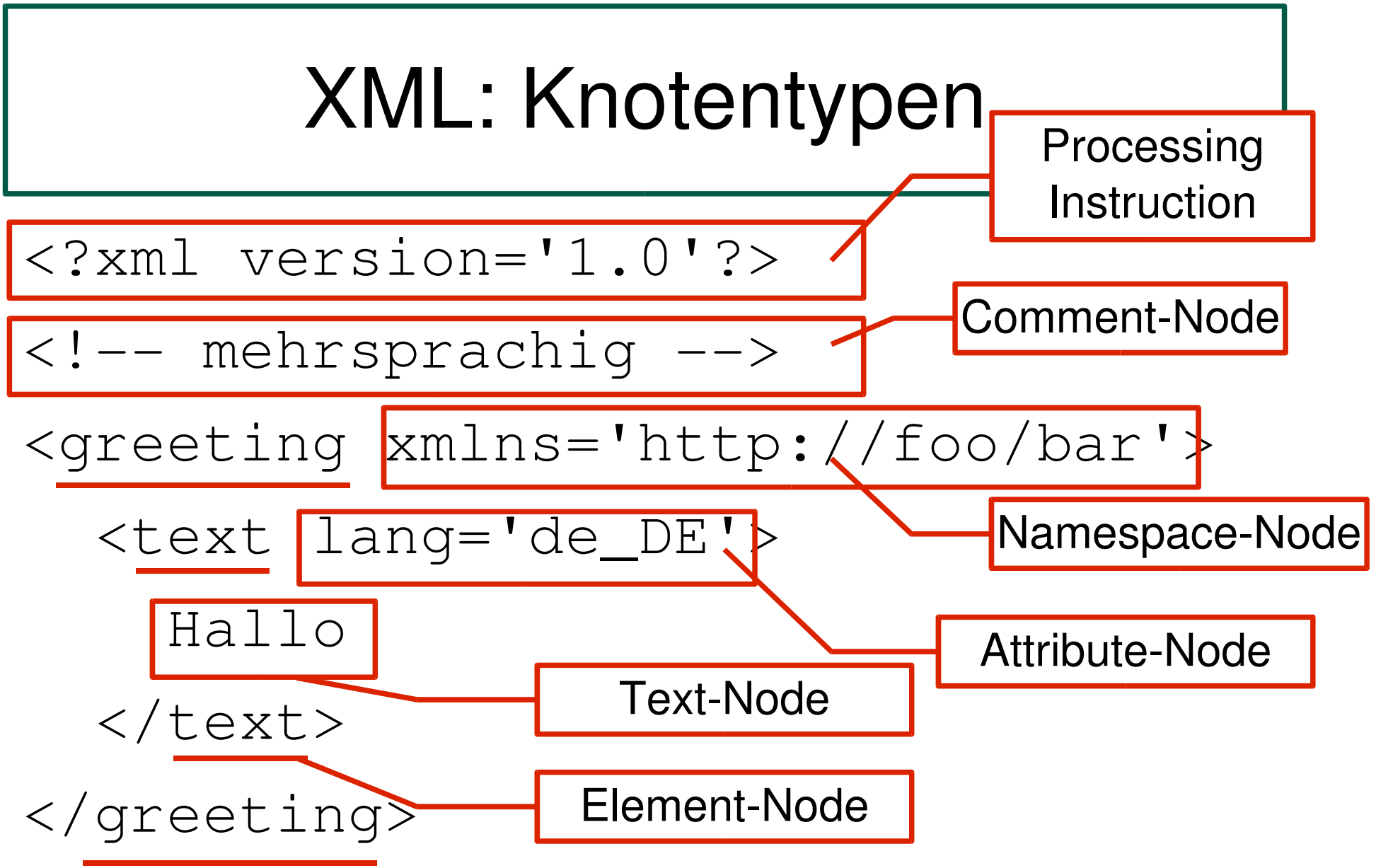
XML-Verarbeitung

- TextTools: sed, awk, grep, ...
- Hochsprache: string-Operationen (z.B. TCL: Iconml)
- API: SAX, DOM
- Spezialsprache: XSLT
  - Eingebettet in Browser (Mozilla, IE)
  - Eigenständiges Programm (xsltproc, xalan)
  - Bibliothek (libxsl, msxml, xalan)

Einordnung in Umfeld

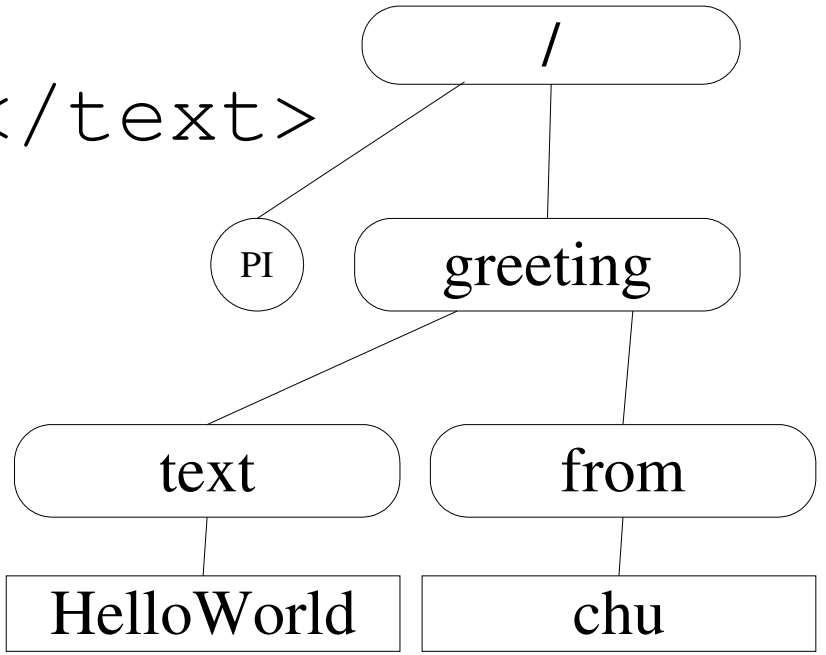
- XML
- XPath
- XSLT
- XSL:FO, XHTML

XML: Knotentypen



HELLO-World

```
<?xml version="1.0"?>
<greeting>
  <text>Hello World</text>
  <from>chu</from>
</greeting>
```



Grundlegende Arbeitsweise

1. Parsen des Stylesheets
2. Parsen des XML-Dokumentes
3. Anwenden des Stylesheets auf das Dokument
4. Serialisierung des Ergebnisses

Erstes XSLT

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>

<xsl:template match="greeting">
  Gesendet wird: <xsl:value-of select="text"/>
  von <xsl:value-of select="from"/>
</xsl:template>

</xsl:stylesheet>
```

- Automatisch: `<xsl:apply-templates select="/"/>`

Anwendung des Stylesheets

- Aktuell bearbeiteter Knoten bildet den *Kontext*
- Rekursive Traversierung des Eingabebaumes
  - Für jeden Knoten im Kontext:
    - Finde das am *besten passende* Template
    - Wende dieses Template auf Knoten an (Kontextwechsel)
- Implizit definierte *Built-In* Templates

Ausgabe

```
$ xsltproc hello.xsl hello.xml
Gesendet wird: Hello World von chu
```

Stylesheet-Aufbau

- xsl:stylesheet
  - xsl:include, xsl:import
  - xsl:param, xsl:variable, xsl:key
  - xsl:output, xsl:strip-space, xsl:preserve-space
  - xsl:attribute-set
  - xsl:template match, xsl:template name

Selecting

```
<xsl:apply-templates select="???">
```

- Achsen
- Pfad
- Prädikate
- Mengenvereinigungen: |

Built-In-Templates

- `xsl:template match="*|/"`
  - `xsl:apply-templates select="."`
- Default-Templates für alle Modi
- `xsl:template match="text()|@"`
  - `xsl:value-of select="."`
- pi- und comment-Templates -> leer

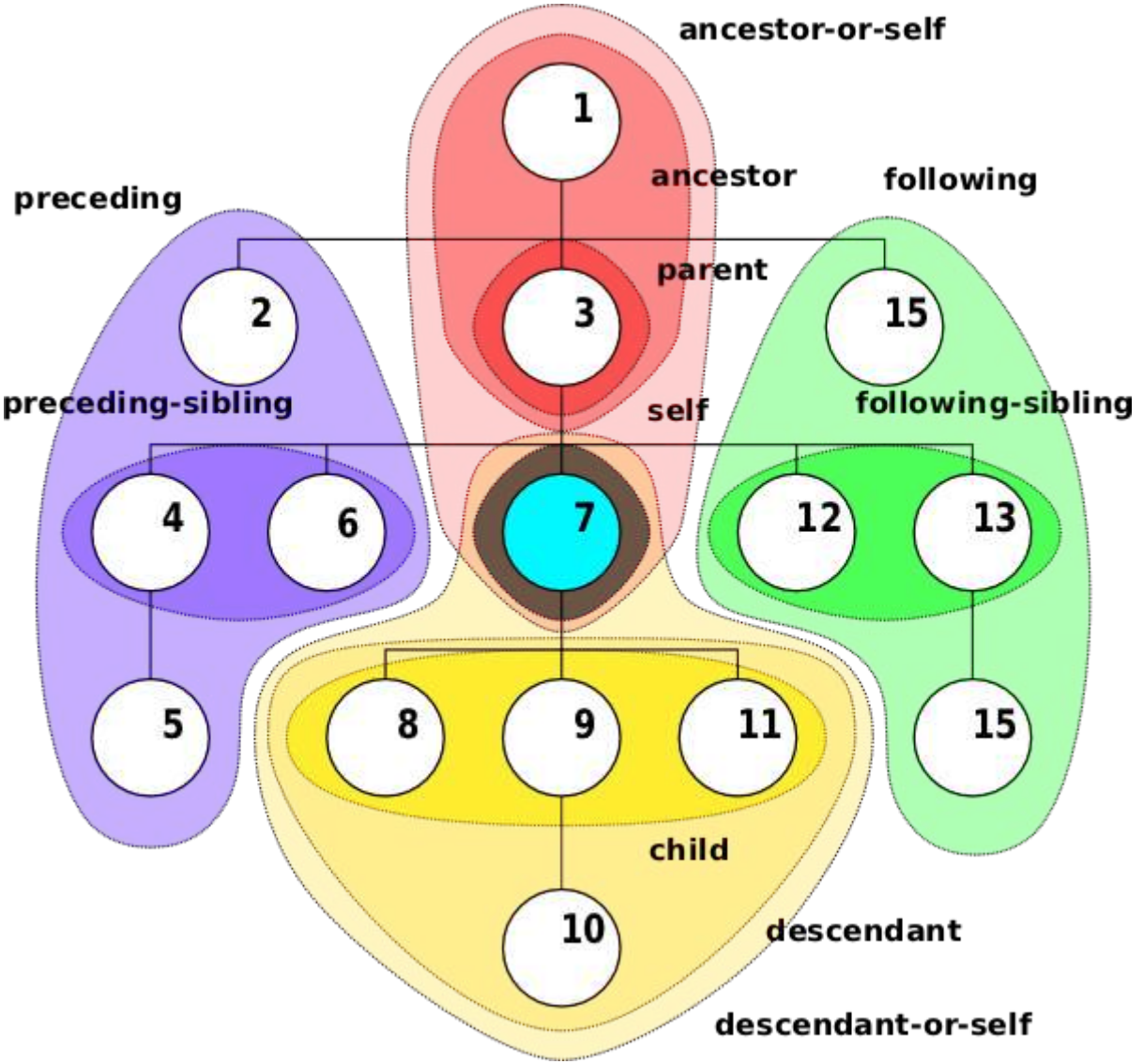
Modi

- 2. Dimension zur Auswahl der Templates
- Nur Templates mit passendem *mode* werden genutzt
- In `xsl:template` und `xsl:apply-templates` anzugeben

Achsen

- Legen Suchrichtung fest oder wählen Knotentypen
- attribute (@), namespace
- Angabe im xpath: `achse::pfad`

Achsen


ancestor-or-self
1
ancestor
following
preceding
parent
2
3
15
preceding-sibling
self
following-sibling
4
6
7
12
13
5
8
9
11
15
child
10
descendant
descendant-or-self

Xpath: Pfade

Analogie: Dateisystem, Trennsymbol: /

- Wurzelknoten: `select='/'`
- Aktueller Knoten: `select='.'`
- Elternknoten: `select='..'`
- Kind: `select='bezeichner'`
- Enkel: `select='bez1/bez2'`
- Urenkel: `select='bez1/bez2/bez3'`

Wildcards

Üblicherweise Matching nach Name

- Beliebige Element-Knoten: `select='*'`
- Beliebige Attribut-Knoten: `select='@*'`
- Knotentext: `select='text()'`
- Beliebige Knotentypen: `select='node()'`
- Beliebige Zwischenknoten: `select='//'`

Prädikate

- In `[]` zur genaueren Auswahl von Knoten
- Aufruf von Funktionen
  - `position(), last(), concat(), substring(), sum(), ...`
- Numerische Vergleiche und Operationen
  - `<, >, =, !=, >=, <=; +,-,*, mod, div`
- Logische Ausdrücke
  - `not(), and, or, true(), false()`

Kontrollfluss

- `<xsl:if test='...'> </xsl:if>`
- ```
  <xsl:choose>
  <xsl:when test='...'> </xsl:when>*
  <xsl:otherwise> </xsl:otherwise>
  </xsl:choose>
  ```
- ```
  <xsl:for-each select='...'>
  </xsl:for-each>
  ```
- `<xsl:apply-templates select='...'/>`
- `<xsl:call-template name='...'/>`

<xsl:call-template> vs. <xsl:apply-template>

- <xsl:apply-template select='Expr1'>
  → <xsl:template match='Expr2'>
  - Dokument gibt Reihenfolge der Abarbeitung vor
    - Flexibel, aber unvorhersehbarer
- <xsl:call-template name='Name'>
  → <xsl:template name='Name'>
  - Style gibt Reihenfolge der Abarbeitung vor
    - Vorhersehbar, aber weniger gut wiederverwendbar

Variable, Parameter

- `<xsl:variable name='...'>`
- Parameter für Templates und Stylesheets
  - `<xsl:param name='...'>`
  - `<xsl:with-param name='...'>`
- Zugriff auf Variablen oder Parameter: `$name`
- Variablen können nicht verändert werden!

  → häufige Verwendung von Rekursion

Kombinieren

- Importieren von weiteren XSL-Dateien mit `xsl:import` oder `xsl:include`
- Nachladen von weiteren XML-Dateien mit `xsl:document()`-Funktion
- `<xsl:apply-templates select="document(bla.xml)">`

Sortieren

- `xsl:sort` als Kind in `xsl:for-each` oder `xsl:apply-templates`
- Knoten werden in Sort-Order abgearbeitet

Weitere Themen

- Verlinkungen und Referenzierungen
- Gruppieren
- Attribute-Sets
- Erweiterungen mit EXSLT
- Erweiterungen durch Programmiersprachen

Literatur

- XSLT, Doug Tidwell, O'Reilly
- XSLT-Cookbook, Sal Mangano, O'Reilly
- http://www.w3.org/Style/XSL/
- http://www.vbxml.com/xsl/tutorials/intro/
- http://www.zvon.org/xxl/XSLTutorial/