XML-Praxis

XSLT – XSL Transformations

Teil 1

Jörn Clausen

joern@TechFak.Uni-Bielefeld.DE

Übersicht

- Namespaces
- XPath
- einfache XSLT-Stylesheets
  - template rules
  - default rules

Namespaces

- XML-Sprachen für wiederkehrende Probleme:
  - Tabellen
  - mathematischer Formelsatz
  - genetische Sequenzen
  - ...
- Kombination/Einbettung von Sprachen
- Beispiel: (X)HTML-Dokument mit Formeln in MathML
- Problem: Was gehört zu welcher XML-Sprache?

Verwendung von Namespaces

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    <p>also sprach Pythagoras:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <msup><mi>x</mi><mn>2</mn></msup>
        <mo>+</mo>
        <msup><mi>y</mi><mn>2</mn></msup>
        ...
    </math>
  </body>
</html>
```

alternative Notation von Namespaces

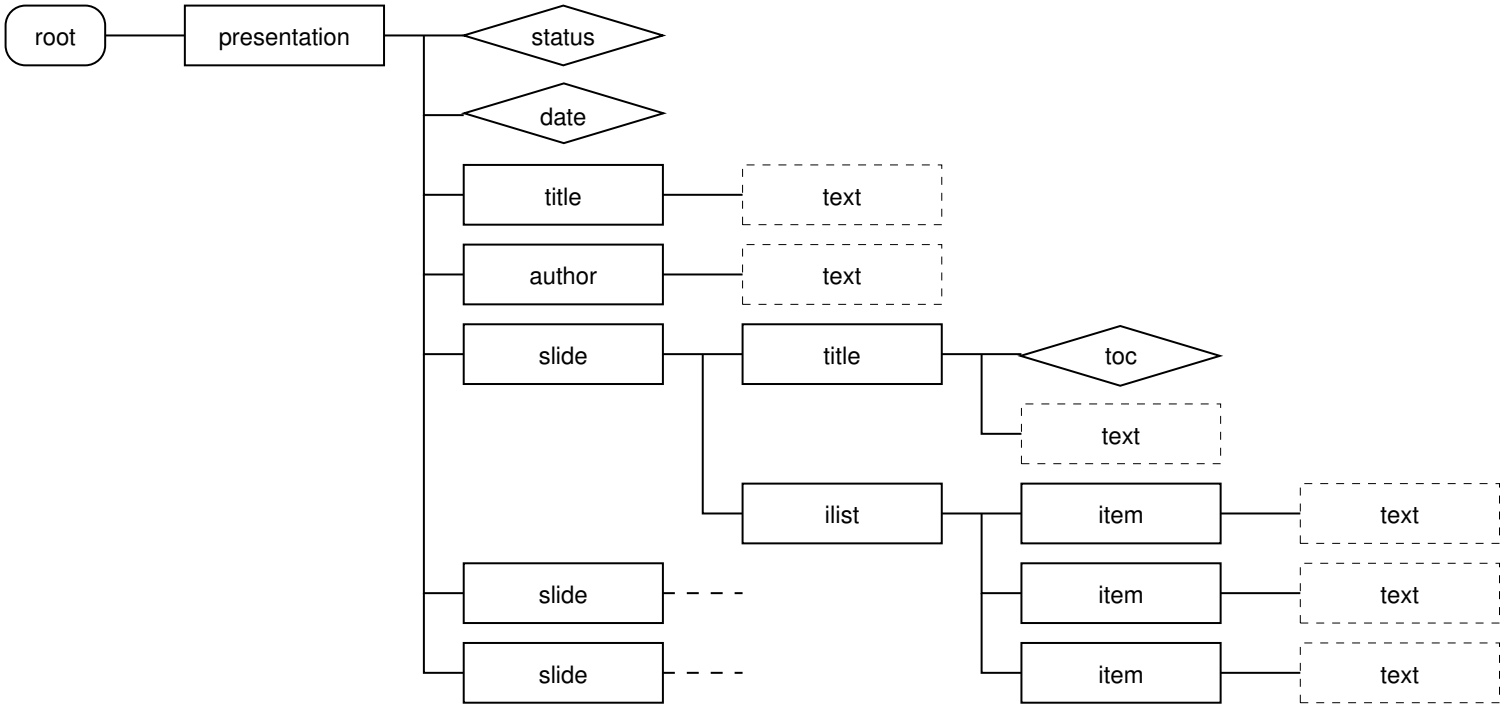
```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ml="http://www.w3.org/1998/Math/MathML">
  <head>...</head>
  <body>
    <p>also sprach Pythagoras:</p>
    <ml:math>
      <ml:mrow>
        <ml:msup><ml:mi>x</ml:mi><ml:mn>2</ml:mn></ml:msup>
        <ml:mo>+</ml:mo>
        <ml:msup><ml:mi>y</ml:mi><ml:mn>2</ml:mn></ml:msup>
        ...
    </ml:math>
  </body>
</html>
```

Namespaces, cont.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ml="http://www.w3.org/1998/Math/MathML">
```

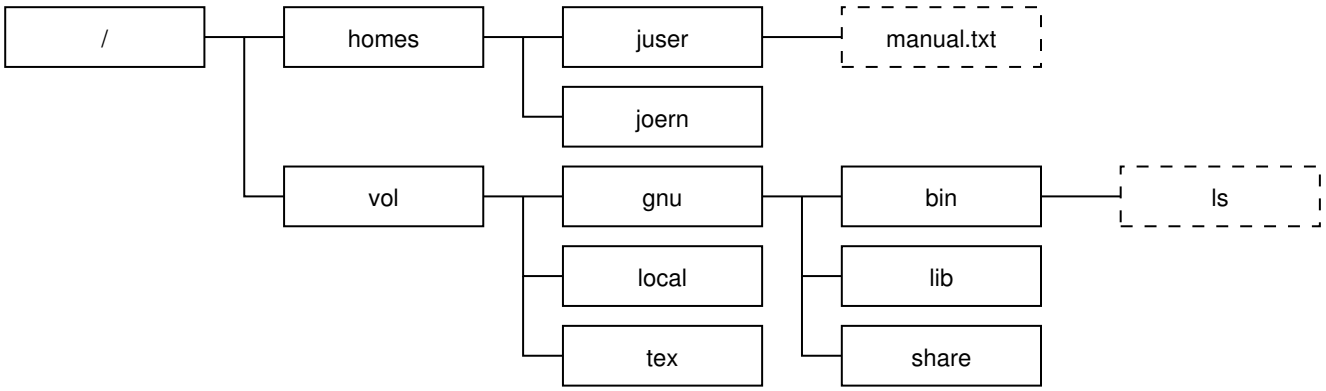
- default namespace
- namespace prefix beliebig
- namespace URI ausschlaggebend, muß exakt übereinstimmen
- keine Verknüpfung mit Grammatik/Schema
- URL als identifier extrem schlechte Wahl
- schlecht mit DTDs zu realisieren

XML-Dokument als Baum



- weitere Text-Knoten durch whitespace
- Aufgabe: lokalisiere einen (oder mehrere) Knoten

Analogie Dateisystem



- absoluter Pfad: Wegbeschreibung vom Wurzelverzeichnis

  `/vol/gnu/bin/ls`

- relativer Pfad: Wegbeschreibung vom „aktuellen“ Verzeichnis

  `../juser/manual.txt`

- Unterschied bei XML: gleichnamige Kindknoten

XPath

- XPath beschreibt Pfade im XML-Baum
- Knotentypen: Elementknoten, Attributknoten, Textknoten, ...
- *Achsen*: child, parent, sibling, descendant, ..., attribute
- Beispiele:

```
slide
slide/title
/presentation/author/text()
/presentation/@status
//slide/title
```

- ausführliche Schreibweise:

```
/child::presentation/attribute::status
```

XPath, cont.

- Bedingungen in XPath-Ausdrücken:

```
/presentation/slide/title[@toc="yes"]
/presentation/slide/[title/@toc="yes"]/ilist
/presentation/slide[position()=2]
/presentation/slide[position()=last()]
```

- Funktionen:

```
count(/presentation/slide)
/presentation/slide[contains(title,"XML")]
```

- arithmetische Funktionen
- String-Funktionen

Eigenschaften von XSL

- Ursprünge:
  - CSS (Cascading Style Sheets)
  - DSSSL (Document Style and Semantics Specification Language)
- auf XML übertragen: XSL (Extensible Stylesheet Language)
- zwei Teile:
  - XSLT (XSL Transformations)
  - XSL-FO (XSL Formatting Objects)

Eigenschaften von XSLT

- Transformation zwischen XML-Bäumen
- vollständige Programmiersprache
- Bezeichnung „style sheet“ hat sich trotzdem gehalten
- deklarativ, ohne Seiteneffekte
- ähnlich zu funktionalen Sprachen (Haskell)
- regelbasiert
- ist selber XML

ein einfaches Stylesheet

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="ISO-8859-1"/>

  <xsl:template match="/presentation">
    <html>
      <head>
        <title>Presentation</title>
      </head>
      <body>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

ein einfaches Stylesheet, cont.

- HTML hinreichend ähnlich zu XML
- Ausgabe:

```
<html>
  <head>
    <meta content="text/html; charset=ISO-8859-1"
          http-equiv="Content-Type">
    <title>Presentation</title>
  </head>
  <body>
  </body>
</html>
```

templates

- XSLT-Prozessor verarbeitet Dokument entsprechend Baumstruktur
- zum aktuellen Knoten passendes *template* wird ausgeführt

```
<xsl:template match="/presentation">
  <html>
    ...
  </html>
</xsl:template>
```

- Text innerhalb des templates wird ausgegeben
- Stylesheet muß wohlgeformt sein
- XSLT-Anweisungen werden verarbeitet

templates kombinieren

- Kontrolle an XSLT-Prozessor zurückgeben

```
<xsl:template match="/presentation">
  <html>
    <head>...<head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

- rekursive Verarbeitung des nächsten Knoten

templates kombinieren, cont.

```
<ilist>
  <item>XML is ...</item>
  <item>XML instances ...</item>
</ilist>
```

templates kombinieren, cont.

```
<ilist>
  <item>XML is ...</item>
  <item>XML instances ...</item>
</ilist>
```

```
<ul>
  <li>XML is ...</li>
  <li>XML instances ...</li>
</ul>
```

templates kombinieren, cont.

```
<ilist>
  <item>XML is ...</item>
  <item>XML instances ...</item>
</ilist>
```

```
<ul>
  <li>XML is ...</li>
  <li>XML instances ...</li>
</ul>
```

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

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

default rules

- template rule für `item`:

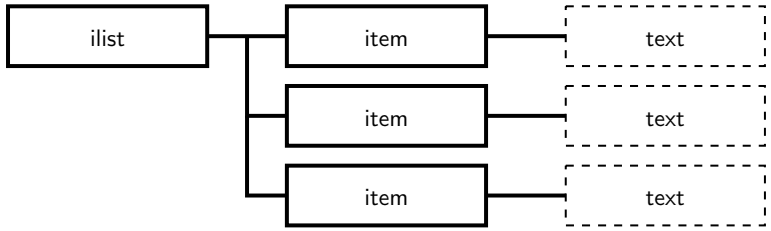
```
<xsl:template match="item">
  <li>
    <xsl:apply-templates/>
  </li>
</xsl:template>
```

- *default rules*:

element/root node	verarbeite Kind-Knoten
text node	gib Text aus
attribute/comment node	ignorieren

- Stylesheets können top-down erstellt werden

der Transformationsprozeß

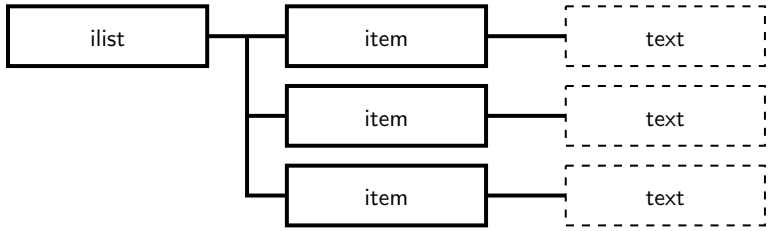


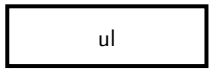

```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß



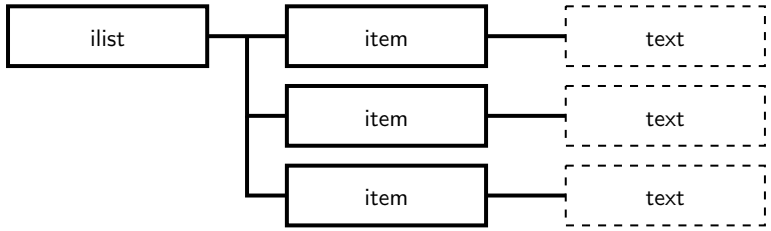


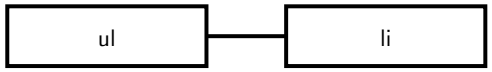

```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß



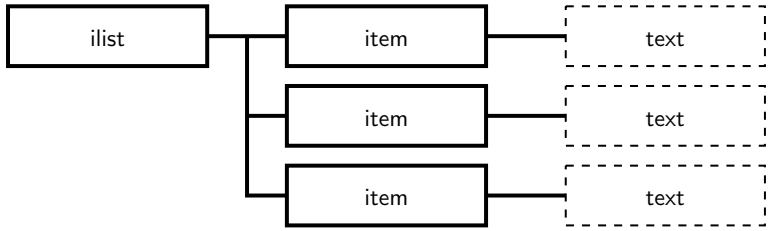


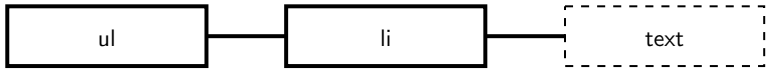

```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß



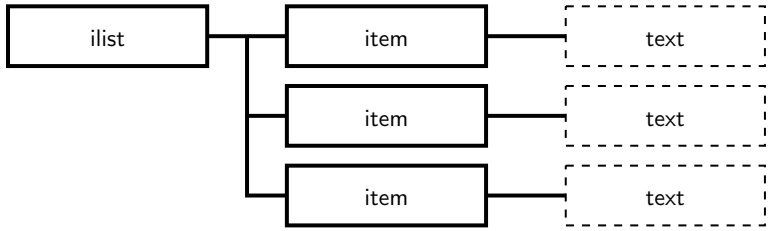



```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß

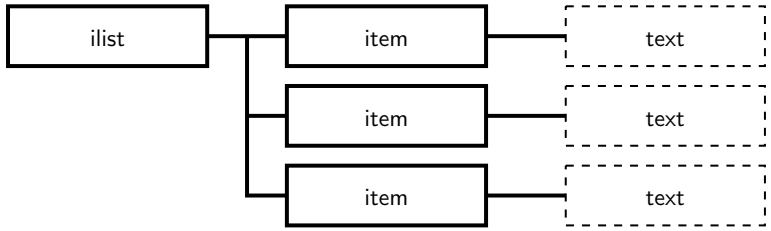



```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß

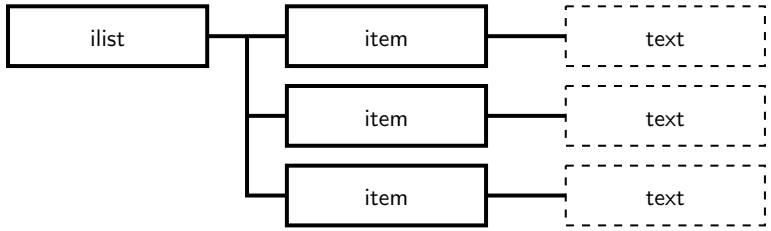


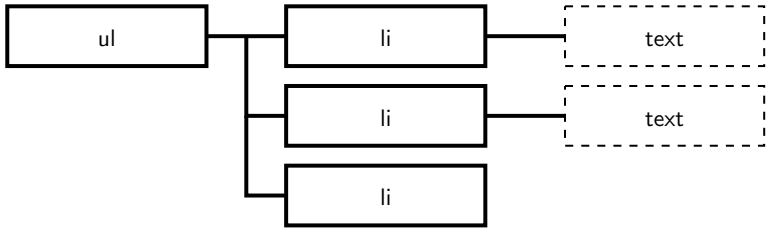

```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß



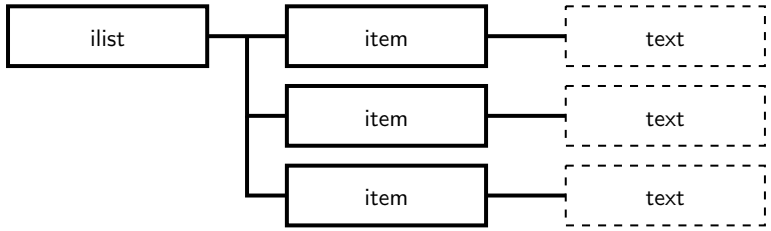


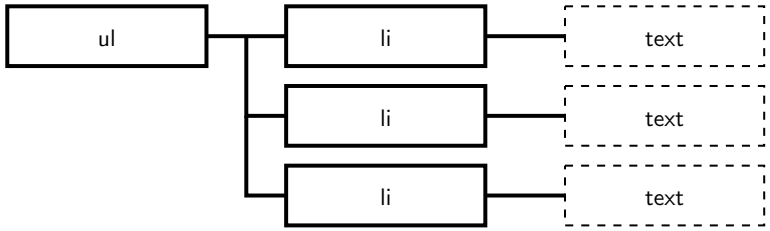

```
<xsl:template match="ilist">
  <ul>
    <xsl:apply-templates/>
  </ul>
</xsl:template>
```

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

- default rule für Text-Knoten

der Transformationsprozeß






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

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

- default rule für Text-Knoten

templates, cont.

- match-Attribut vollständiger XPath-Ausdruck:

```
<xsl:template match="presentation/title">
  <h1>
    <xsl:apply-templates/>
  </h1>
</xsl:template>

<xsl:template match="slide/title">
  <h2>
    <xsl:number count="slide"/>: <xsl:apply-templates/>
  </h2>
</xsl:template>
```

Daten auswählen

- Verarbeitung in Dokumentreihenfolge nicht immer sinnvoll
- HTML-Seite mit Titel:

```
<xsl:template match="/presentation">
  <html>
    <head>
      <title>
        <xsl:value-of select="title"/>
      </title>
    </head>
    ...
</xsl:template>
```

- XPath-Pfad vom aktuellen Knoten aus

Daten auswählen, cont.

- Navigationselemente: vorherige Folie

```
<xsl:value-of select="preceding-sibling::slide[1]/title"/>
```

- `preceding`/`preceding-sibling` liefern Knoten in umgekehrter Dokumentreihenfolge
- analog nächste Folie

```
<xsl:value-of select="following-sibling::slide[1]/title"/>
```