Orientation in Objects

XSLT 2.0 verstehen

Was ist neu in XSLT 2.0?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.1

Who we are

Orientation in Objects

Tobias Kieninger

Matthias Born

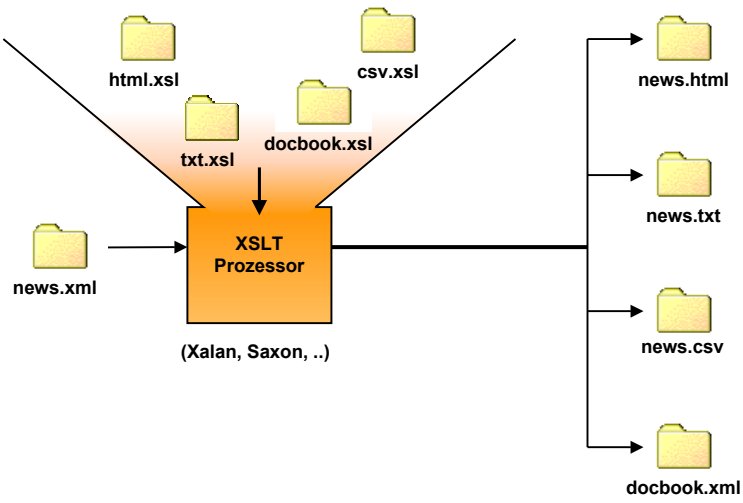
XSLT 2.0 verstehen

Orientation in Objects

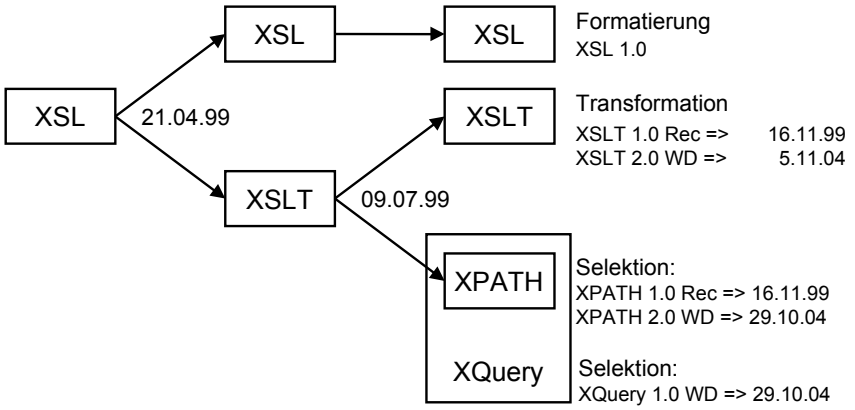
- **Motivation**
- Konzeptionelle Änderungen
- Neue Elemente
- Mehrere Ausgabeformate
- Reguläre Ausdrücke
- Datentypen / XML-Schema
- Erweiterte XPath Ausdrücke...

XSLT allgemein

Orientation in Objects



Von XSL zu XSLT



Spezifikationen und Prozessoren

- Spezifikationen
  - XSLT 2.0: W3C Working Draft 5 Nov. 2004
  - XPath 2.0: W3C Working Draft 29 Oct. 2004
  - XQuery 1.0: W3C Working Draft 29 Oct. 2004
- Prozessoren (XSLT 2.0)
  - Saxon von Michael Kay „The XSLT and XQuery Processor “
    - **Basic XSLT Processor (frei)**
    - **Schema Aware XSLT Processor (kommerziell)**
  - Altova (XMLSpy)
    - **Basic XSLT Processor**

XSLT 2.0 verstehen

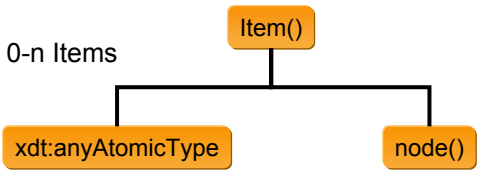
Orientation in Objects

- Motivation
- **Konzeptionelle Änderungen**
- Neue Elemente
- Mehrere Ausgabeformate
- Reguläre Ausdrücke
- Datentypen / XML-Schema
- Erweiterte XPath Ausdrücke...

Sequenzen

Orientation in Objects

- XPath 2.0 Ausdrücke: **Sequenzen** anstatt **NodeSets**
  - Definierte Reihenfolge
  - Können Duplikate enthalten
  - Können heterogen sein
- Sequenzen bestehen aus 0-n Items
- 2 Typen von Item
  - Simple Typed Values
  - Knoten



- /books/book => Sequenz aus Elemente book in Document Order
- (1, 2, 3, ‘eins‘, ‘zwei‘, @id, foo)

Wortkunde

```
<xsl:stylesheet version="2.0"  ... >

 <xsl:output method="xml" version="1.0"/>

  <xsl:template match="cd">
    ... <h1>
    <xsl:apply-templates/>
    ... </h1>
  </xsl:template>

</xsl:stylesheet>
```

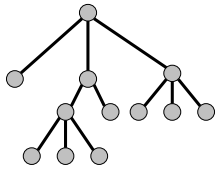
Declaration

Sequence Constructor

Instruction

Temp. Bäume ersetzt das RTF

```
<xsl:variable name="foo" select="/book"/>
<xsl:variable name="bar">
     <book>
         <chapter>
           <para>foo</para>
           <para>bar</para>
         </chapter>
     </book>
 </xsl:variable>
```



2 Variablen, 2 Aufrufe, 2 XSLT Versionen?!

```
<xsl:for-each select="$foo//para">
<xsl:for-each select="$bar//para">
```

Tunnelparameter

Orientation in Objects

- Tunnelparameter
  - <xsl:with-param tunnel="yes|no"/>
- Genehmigung, im Tunnelstrom zu fischen
  - <xsl:param tunnel="yes|no"/>
- Build-In Templates tunneln in jedem Fall
  - auch ohne die Angabe von tunnel=„yes“

XSLT 2.0 verstehen

Orientation in Objects

- Motivation
- Konzeptionelle Änderungen
- **Neue Elemente**
- Mehrere Ausgabeformate
- Reguläre Ausdrücke
- Datentypen / XML-Schema
- Erweiterte XPath Ausdrücke...

Neue Elemente in XSLT 2.0

Orientation in Objects

12 + 3 Declarations (TL-Elemente)

- <xsl:character-map>
  - <xsl:output-character>
- <xsl:function>
- <xsl:import-schema>

21 + 11 Instructions (Befehle)

- <xsl:analyze-string>
  - <xsl:matching-substring>
  - <xsl:non-matching-substring>
- <xsl:document>
- <xsl:for-each-group>
- <xsl:namespace>
- <xsl:next-match>
- <xsl:perform-sort>
- <xsl:result-document>
- <xsl:sequence>
- <xsl:output-character>

<xsl:value-of/> in XSLT 1.0 und 2.0

Orientation in Objects

```
<xsl:variable name="test">
  <test>
    <item>Item 1</item>
    <item>Item 2</item>
    <item>Item 3</item>
  </test>
</xsl:variable>
```

XSLT 1.0:
Item 1

XSLT 2.0:
Item 1 Item 2 Item 3

```
<xsl:template match="test">
  <!-- XSLT 1.0 -->
  <xsl:value-of select="$test//item" version="1.0"/>
  <!-- XSLT 2.0 -->
  <xsl:value-of select="$test//item"/>
</xsl:template>
```

<xsl:value-of>

Orientation in Objects

- **Neu:** Attribut *separator*
  - Trennzeichen für die einzelnen Elemente einer Sequenz
- Attribut *select* kann entfallen, statt dessen Sequenzkonstruktor

```
<xsl:value-of>
  <xsl:for-each select="(1 to 10)">
    <xsl:text>Hello World-Call #</xsl:text>
      <xsl:value-of select="."/>
  </xsl:for-each>
</xsl:value-of>
```


<xsl:sequence/>

Orientation in Objects

- Bildet Sequenz über Werte oder Knoten

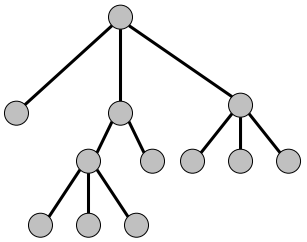
```
<xsl:variable name="values" as="xs:integer*">
   <xsl:sequence select="(1,2,3,4)"/>
   <xsl:sequence select="@price"/>
</xsl:variable>

<xsl:value-of select="sum($values)"/>
```

Gruppieren in XSLT 1.0 und 2.0 - Die Quelle

Orientation in Objects

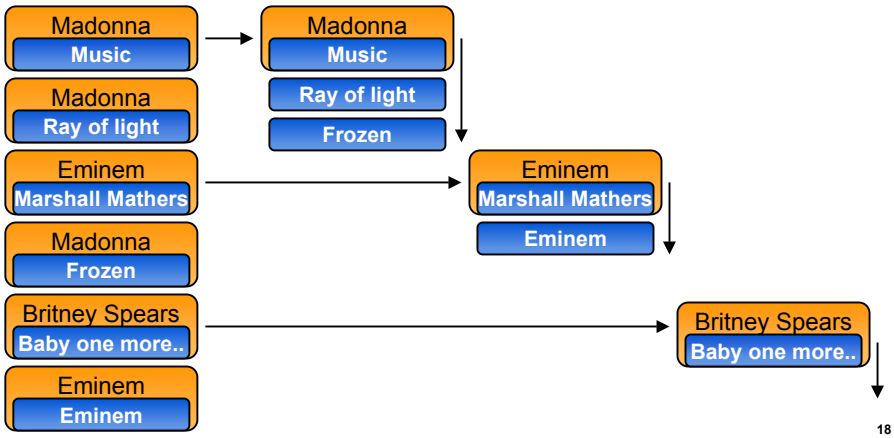
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cdliste>
  ..
  <cd id="65" jahr="2000">
    <titel>Music</titel>
    <interpret>Madonna</interpret>
    <hersteller>BMG</hersteller>
    <preis/>
    <track id="01">
      <titel>Music</titel>
      <laenge>4:35</laenge>
    </track>
    <track id="02">
      <titel>Impressive Instant</titel>
      <laenge>5:10</laenge>
    </track>
  </cd>
...
</cdliste>
```



Gruppieren in XSLT 1.0

Orientation in Objects

```
<xsl:for-each select="//cd[generate-id(.)=
                      generate-id(key('cds',interpret)[1])]">
    <xsl:for-each select="key('cds', interpret)">
```



Gruppieren nach Muench: Beispiel

Orientation in Objects

```
<xsl:key name="cds" match="cd" use="interpret"/>

<xsl:for-each select="//cd[generate-id(.) =
                          generate-id(key('cds',interpret)[1])]">
   <xsl:value-of select="interpret"/>
   <br/>

   <xsl:for-each select="key('cds', interpret)">
        Titel: <xsl:value-of select="titel"/>
        <br/>
        Hersteller: <xsl:value-of select="hersteller"/>
        <br/>
   </xsl:for-each>
   <br/>
</xsl:for-each>
```


Simplified Grouping <xsl:for-each-group/>

Orientation in Objects

```
<xsl:for-each-group select="cdliste/cd"
                    group-by="interpret">
   <xsl:value-of select="current-grouping-key()"/>
   <br/>
    <!-- <xsl:value-of select="interpret"/> -->
   <xsl:for-each select="current-group()">
        Titel: <xsl:value-of select="titel"/>
        <br/>
        Hersteller: <xsl:value-of select="hersteller"/>
        <br/>
   </xsl:for-each>
   <br/>
</xsl:for-each-group>
```

group-starting-with: HTML nach Docbook

Orientation in Objects

Quelle:

```
<h1>Überschrift</h1>
<p>Absatz</p>
<p>Absatz</p>
<h1>Überschrift</h1>
<p>Absatz</p>
<p>Absatz</p>
```

Ziel:

```
<section>
  <title>Überschrift</title>
  <para>Absatz</para>
  <para>Absatz</para>
</section>
<section>
  <title>Überschrift</title>
  <para>Absatz</para>
  <para>Absatz</para>
</section>
```


group-starting-with: HTML nach Docbook

Orientation in Objects

Stylesheet:

```
<xsl:for-each-group select="*" group-starting-with="h1">
  <section>
    <title>
      <xsl:value-of select="current-group()[self::h1]"/>
    </title>
    <xsl:for-each select="current-group()[self::p]">
      <param><xsl:apply-templates/></param>
    </xsl:for-each>
  </section>
</xsl:for-each-group>
```

Gruppiermöglichkeiten <xsl:for-each-group>

Orientation in Objects

Sequenz: A B B A C C B B A C

- **group-by**
  - AAA | BBBB | CCC
- **group-adjacent**
  - A | BB | A | CC | BB | A | C
  - Gruppiert werden auf einander folgende Items
- **group-starting-with=„A“**
  - ABB | ACCBB | AC
- **group-ending-with=„A“**
  - A | BBA | CCBBA | C

Stylesheet Funktionen

Orientation in Objects

- Benutzerdefinierte Funktionen,
- Kann in XPath Ausdrücken verwendet werden
- Wiederverwendung und Modularität von Stylesheets

```
<xsl:function name="oio:round" as="xs:integer">
   <xsl:param name="wert" as="xs:double"/>
   <xsl:sequence select="round($wert div 10) * 10"/>
</xsl:function>

<xsl:value-of select="oio:round(212)"/>
```

<xsl:next-match/> Beispiel

Orientation in Objects

Aufruf der **nächsten** Template Rule des **selben Elementes** mit **geringerer Priorität**

```
<xsl:template match="node[@prio='highest']">
  <span style="background-color:red">
    <xsl:next-match/>
  </span>
</xsl:template>

<xsl:template match="node">
 <b>
   <xsl:text>Beipiel: </xsl:text>
   <xsl:apply-templates/>
 </b>
</xsl:template>
```

<xsl:character-map/>

Orientation in Objects

```
<xsl:character-map name="all" use-character-maps="o,u">
   <xsl:output-character character="Ä" string="Ae"/>
   <xsl:output-character character="ä" string="ae"/>
   <xsl:output-character character="ß" string="ss"/>
</xsl:character-map>

<xsl:variable name="text" as="xs:string*">
   <xsl:sequence select="('Ä','ä')"/> ...
   <xsl:sequence select="('ß')"/>
</xsl:variable>

<xsl:output method="text" use-character-maps="all"/>

<xsl:value-of select="$text" separator=","/>
```

<xsl:perform-sort/>

- Sortieren beliebiger Sequenzen
- Sortieralgorithmus definiert durch 1-n <xsl:sort> Elemente
- Sequenz wird in Variable gespeichert oder als Ergebnis einer Funktion zurück gegeben
- Vorteil: Elemente werden nicht abgearbeitet

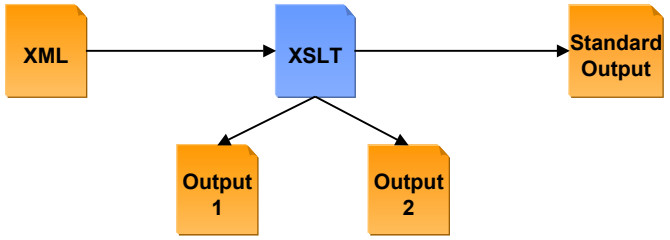
```
<xsl:variable name="sorted-cds“ as="xs:string“ >
    <xsl:perform-sort select="//cd">
        <xsl:sort select="xs:string(interpret)"/>
        <xsl:sort select="xs:integer(@jahr)"/>
    </xsl:perform-sort>
</xsl:function>
```


XSLT 2.0 verstehen

- Motivation
- Konzeptionelle Änderungen
- Neue Elemente
- **Mehrere Ausgabeformate**
- Reguläre Ausdrücke
- Datentypen / XML-Schema
- Erweiterte XPath Ausdrücke...

Erzeugen mehrerer Ausgabedateien

- XSLT 1.0 kein Befehl enthalten, Realisierung durch proprietäre Erweiterungen der Prozessoren
  - <saxon:document>
  - <redirect:write> Xalan
- XSLT 1.1 mittels `<xsl:document href="...">`
- XSLT 2.0 mittels `<xsl:result-document href="...">`



Erzeugen mehrerer Outputfiles

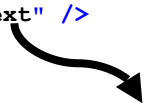
```
<xsl:template match="/">
  <messages>
    <message>Standard-Output</message>
     <xsl:result-document href="output1.html">
      <html>
        <body>Output 1</body>
      </html>
     </xsl:result-document>
     <xsl:result-document href="output2.html">
      <html>
       <body>Output 2</body>
      </html>
     </xsl:result-document>
  </messages>
</xsl:template>
```

Dokument Serialisierung <xsl:output/>

Orientation in Objects

- Definition verschiedener Ausgabeformate
- Unterscheidung durch Attribut ***name***
- Beim Serialisieren können Zeichen gemappt werden
  - z.B. „ä“ zu „ae“
  - Siehe **use-character-maps**
- method
  - **xhtml** | html | xml | text

```
<xsl:output   method="text" use-character-maps="all"
              name="text" />
```



```
<xsl:result-document format="text" href="{$url}" >
```


XSLT 2.0 verstehen

Orientation in Objects

- Motivation
- Konzeptionelle Änderungen
- Neue Elemente
- Mehrere Ausgabeformate
- **Reguläre Ausdrücke**
- Datentypen / XML-Schema
- Erweiterte XPath Ausdrücke...

Wörter eines Absatz zählen (XSLT 1.0)

Orientation in Objects

```
<xsl:call-template name="count">
   <xsl:with-param name="text" select="concat(normalize-space(.),' ')"/>
</xsl:call-template>
<!-- -->
<xsl:template name="count">
   <xsl:param name="text"/>
   <xsl:param name="zahl" select="1"/>
   <xsl:variable name="rest" select="substring-after($text,' ')"/>
   <xsl:choose>
     <xsl:when test="$rest">
       <xsl:call-template name="count">
         <xsl:with-param name="text" select="$rest"/>
         <xsl:with-param name="zahl" select="$zahl + 1"/>
       </xsl:call-template>
     </xsl:when>
     <xsl:otherwise>
       <xsl:value-of select="$zahl"/>
     </xsl:otherwise>
   </xsl:choose>
 </xsl:template>
```


Regular Expressions

Orientation in Objects

- Als Funktionen
  - **tokenize()** Zerlegt einen String in Substrings an der Stelle, die die RegEx beschreibt
  - **matches()** Testet, ob ein String eine best. RegEx entspricht
  - **replace()** Ersetzt den String, beschrieben durch die RegEx, durch einen anderen
  - In Prädikaten:
    - book [**matches (author, '^[A-H].*ss[a-z]+')**]
- Neues Element:
  - <xsl:analyze-string/>

Wörter eines Absatz zählen (XSLT 2.0, tokenize())

Orientation in Objects

```
<xsl:variable name="lorem">Lorem ipsum ... est
   laborum.</xsl:variable>

<xsl:value-of
   select="count(tokenize($lorem,' '))"/>
```

- Schneller (1/3, getestet am Neuen Testament)
- Kürzer und damit weniger anfällig für Fehler

<xsl:analyze-string/>

Orientation in Objects

- Analysieren von Zeichenketten
- Zerlegen der Zeichenkette durch reguläre Ausdrücke

```
<body><p>
   <xsl:analyze-string
        select="unparsed-text('text.txt','UTF-8')"
        regex="\n">
        <xsl:matching-substring>
              <br/>
        </xsl:matching-substring>
        <xsl:non-matching-substring>
              <xsl:value-of select="."/>
        </xsl:non-matching-substring>
   </xsl:analyze-string>
</p></body>
```

Funktion regex-group()

- In den RegEx können Subpattern () verwendet werden
- Über regex-group(n) kann auf die Subpattern 1-n zugegriffen werden
- **Muster:** „Beliebiger String mit Zeichenfolge 47.11 “

```
<xsl:analyze-string select="$src"
                    regex="(\d\d)\.(\d\d)">
   <xsl:matching-substring>
        Zahl 1: <xsl:value-of select="regex-group(1)"/>
        Zahl 2: <xsl:value-of select="regex-group(2)"/>
   </xsl:matching-substring>
```

Ergebnis: Zahl 1: 47
Zahl 2: 11

Funktion unparsed-text()

```
<xsl:variable name="text"
      select="unparsed-text('text.txt','UTF-8')"/>
```

- Zugriff auf unstrukturierte (nicht XML!!!) Dokumente
- Rückgabe ist String
- Weiter verarbeiten mit RegEx

Date Time Formatierung in XSLT 2.0

Orientation in Objects

- **format-date()**
  - format-date(xs:date(‚2004-11-23'),'[D] [MNn] [Y]')
    - **Ausgabe: 23 November 2004**
- **format-time()**
  - format-time(xs:time(‚15:30:00'),'[h].[m01] [Pn]')
    - **Ausgabe: 3:30 p.m.**
- **format-dateTime()**
  - format-dateTime($dt , "[D].[M].[Y] [H]:[m] ")
    - **Ausgabe: 23.11.2004 15:30**
- Überwiegende Anzahl neuer Funktionen => XPath 2.0

XSLT 2.0 verstehen

Orientation in Objects

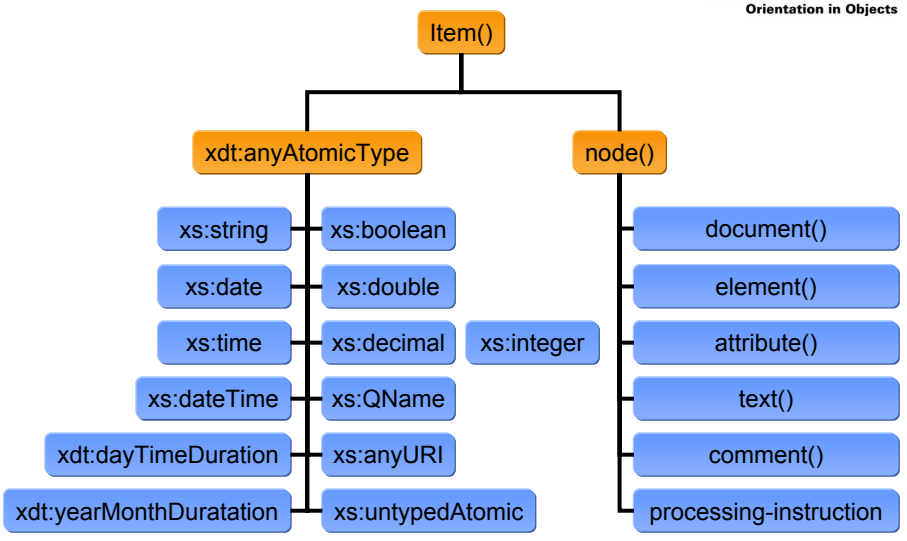
- Motivation
- Konzeptionelle Änderungen
- Neue Elemente
- Mehrere Ausgabeformate
- Reguläre Ausdrücke
- **Datentypen / XML-Schema**
- Erweiterte XPath Ausdrücke...

Typisierung in XSLT 1.0

- Node-Set
- String
- Boolean
- Number
- Result-Tree-Fragment

- Values has always been typed in XSLT: strings, numbers, booleans

Build-In Datentypen (Basic XSLT Processor)



Typisierung in XSLT 2.0

Orientation in Objects

- Basierend auf XML Schema Datentypen
- Build-In: 19 Simple Types
  - dates, years, months, URI etc.
- XSLT Prozessor: effizienterer Code
- Weniger anfällig für Fehler
  - Bsp: Sortieren von 12,50 und 2,25

```
<xsl:param name="wert" as="xs:double"/>
```


Typisierung

Orientation in Objects

- Typzuweisung mittes Attribut @as von
  - <xsl:variable>
  - <xsl:param>
  - <xsl:function>
  - <xsl:template>
- Explizite Typkonvertierung (Konstruktoren) möglich
  - xs:date()
  - xs:integer()

```
<xsl:variable name="values" as="xs:integer*">
   <xsl:sequence select="(1,2,3,4)"/>
   <xsl:sequence select="xs:integer(@price)"/>
</xsl:variable>
```

XSLT und XML Schema

Orientation in Objects

- Menge von „Regeln“, der ein XML Dokument entsprechen muss
- Schema Unterstützung ist optional

- Unterstützte Datentypen:
  - Basic Processor: 19 Datentypen
  - Schema Aware Processor
    - **> 40 Datentypen**
    - **Benutzerdefinierte Datentypen**

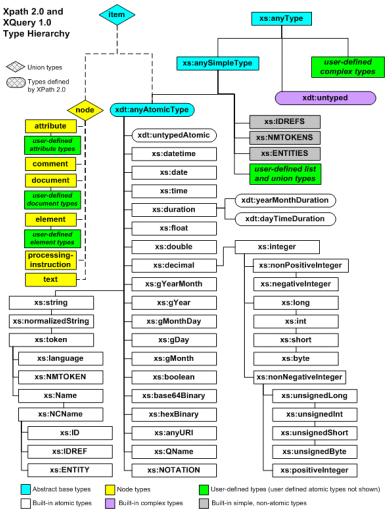
Validierung mittels Schema

Orientation in Objects

- Validierung möglich
  - Quelle vor der Bearbeitung
  - Elemente und Attribute während der Transformation
  - Temporäre Bäume
  - Result-Document vor dem Serialisieren

Datentypen Schema Aware XSLT Processor

Orientation in Objects



- Alle XML Schema Datentypen
- „User Defined“ Datentypen

(Quelle: http://www.w3.org/TR/xpath-datamodel/)

Benutzerdefinierte Datentypen verwenden

Orientation in Objects

Schema:

```
<xs:complexType name="amount">
  <xs:simpleContent>
    <xs:extension base="xs:decimal">
      <xs:attribute name="currency">
        ...
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Stylesheet:

```
<xsl:template match="element( * , amount )">
   <xsl:value-of select="@currency, format-number(., '#,##0.00')"/>
</xsl:template>
```

<xsl:document/>

Orientation in Objects

- Entspricht **nicht** <xsl:document/> aus XSLT 1.1
- Temporäres Dokument für document-level Validierung
- <xsl:document/> != <xsl:result-document/>

```
<xsl:variable name="temp">
   <xsl:document type="mf:invoiceType">
        <invoice>
              <xsl:call-template name="build-invoice"/>
        </invoice>
   </xsl:document>
</xsl:variable>
```


XSLT 2.0 verstehen

Orientation in Objects

- Motivation
- Konzeptionelle Änderungen
- Neue Elemente
- Mehrere Ausgabeformate
- Reguläre Ausdrücke
- Datentypen / XML-Schema
- **Erweiterte XPath Ausdrücke**...

Übersicht XPath 2.0

Orientation in Objects

- Die Akzeptanz einer Programmiersprache steht und fällt mit dem Umfang ihrer internen Funktionsbibliothek
- „Expression language for processing sequences“
- XPath 2.0 ist keine eigenständige Sprache mehr, sondern Teilmenge von XQuery (SQL für XML)
- Streng typisiert durch 45 unterschiedliche Typen (XML Schema)

Übersicht XPath 2.0

Orientation in Objects

- Kommentare in XPath-Anweisungen
  - `(: Kommentar :)`
  - `<xsl:value-of select="$books/(: alle mgl. Elemente :)/author"/>`
- Konstruktor-Funktionen
  - Für Build-In Datentypen
    - **xs:float()**
    - **xs:date()**
    - **xs:integer()**
  - Für User-Defined Atomic-Types
- Benutzerdefinierte Funktionen (<xsl:function>)
  - `<xsl:for-each select="oio:cd-sort(//cd[preis])">`

Neue Operatoren in XPath 2.0

Orientation in Objects

- Schleifen:
  - *for ... in ... return*
- Conditionals
  - *if ... then ... else*
- Bereichsoperator **to**
  - <xsl:for-each select=„(1 to 100)“> 1 2 3 ... 99 100
- Quantifizierende Operatoren (some|every in satisfies)
  - some $x in //name satisfies $x=„Fred“
  - every $x in //name satisfies $x=„Fred“

Neue Operatoren in XPath 2.0

Orientation in Objects

- Mengenoperatoren
  - union, intersect, except
- Vergleichsoperatoren
  - =, !=, <, >, <=, =>, eq, ne, lt, gt, le, ge, is, isnot
- Typoperatoren
  - *cast as, castable as, treat as, instance of*
- Knoten Vergleiche
  - Position: << | >>
  - Identität: is

Conditional Expression: if then else

Orientation in Objects

```
<xsl:value-of select="
   if  @val
   then format-number(@val,'##0.00')
   else 'N/A'"/>
```

- Intuitiver als <xsl:choose>
- Fehler toleranter als <xsl:choose>

Sequenz Expression: for in return

Orientation in Objects

Quelle:

```
<item id="01" name="eins">
   <quantity>2</quantity>
   <price>99.95</price>
</item>
...
```

Summe (aus mehreren Items: 1794,40):

```
<xsl:value-of select="
   sum(for $i
       in //item
       return xs:decimal($i/price) *
                   xs:integer($i/quantity))"/>
```

Vereinigungen, Schnittmengen, Differenzen

Orientation in Objects

- **union** (|)
  - select=„list/**(** cd | dvd | video**)**/title
- **intersect**
  - exists (. intersect $vari)
- **except**
  - Alle Attribute außer ID
  - @* except @id

- Ist der aktuelle Knoten in einer Menge von Knoten enthalten?
  - alt: /foo/bar[gen-id(.)=gen-id($x)]
  - neu: $x intersect /foo/bar
- Alle Knoten auswählen, mit Ausnahme von...
  - alt: /foo/bar[not(gen-id(.)=gen-id($x))
  - neu: /foo/bar except $x

Neue Funktionen in XPath (Auszüge)

Orientation in Objects

- Numerisch
  - abs(), avg(), min(), max()
  - round-half-to-even(), round()

- String
  - RegEx: matches(), replace(), tokenize()
  - string-join()
  - codepoints-to-string(), string-to-codepoints()

- Datum und Zeit
  - current-date(), current-time()
  - day-from-date(), month-from-date()

Neue Funktionen in XPath (Auszüge)

Orientation in Objects

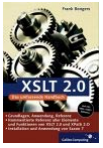
- Sequenzen
  - distinct-values(), exists(), index-of(), insert-before(), deep-equal()
- Eigenschaften von Knoten
  - data(), document-uri(), node-name()
- und und und...

Update zu XSLT 2.0?

Orientation in Objects

- Relativ stabil seit Ende 2003,
  - Recommendation voraussichtlich Anfang 2005
- Update zu XSLT 1.0 mit „geringen“ Änderungen
- Neuerungen sind schnell zu erlernen
- Viele proprietäre Erweiterungen entfallen => Portabilität
- 2 Prozessoren bereits am Markt (Altova, Saxon)
- Typsicherheit: Höhere stabilität Verbessertes Stringhandling (tokenize())

Literaturhinweise



- XSLT 2.0 Programmer's Reference
  Autor: MichSprache: Englisch
  Broschiert - 960 Seiten - Wrox Press
  Erscheinungsdatum: 2. August 2004
  Auflage: 3rd
  ISBN: 0764569090
- XPath 2.0 Programmer's Reference
  Sprache: Englisch
  Broschiert - 552 Seiten - Wrox Press
  Erscheinungsdatum: 1. August 2004
  ISBN: 0764569104
- XSLT 2.0 - Das umfassende Handbuch
  Sprache: Deutsch
  Gebundene Ausgabe - Galileo Press
  Erscheinungsdatum: Juni 2004
  ISBN: 3898423611

Links

- XSL Transformations (XSLT) Version 2.0
  – http://www.w3.org/TR/xslt20/
- XML Path Language (XPath) 2.0
  – http://www.w3.org/TR/xpath20/
- XQuery 1.0 and XPath 2.0 Data Model
  – http://www.w3.org/TR/xpath-datamodel/
- XQuery 1.0 and XPath 2.0 Functions and Operators
  – http://www.w3.org/TR/xpath-functions/
- Google „sequence xslt for-each-grout result-document„

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.1

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.1